

Detecting Contradictions Within and Across Documents

SHENGLIN CHEN, College of Computer Science and Technology, National University of Defense Technology, China

WENFEI FAN, University of Edinburgh, United Kingdom, Shenzhen Institute of Computing Sciences, China, and Beihang University, China

RUOCHUN JIN*, College of Computer Science and Technology, National University of Defense Technology, China

KEXIN MA, College of Computer Science and Technology, National University of Defense Technology, China

This paper presents DocChecker, a system for detecting textual contradictions. Given a document collection $\mathcal{D}$, DocChecker identifies conflicting properties or relationships assigned to the same entity, both within and across documents. It adopts a graph-based paradigm: entities and their relations are extracted, represented as a property graph G , joined across documents, and enriched with domain knowledge. DocChecker introduces a class of rules that detect vertex- and edge-level contradictions in PTIME, unifying logical reasoning, ML predictions and NLP tools. We present how graphs are extracted, joined and enriched, and propose a parallel rule discovery algorithm with new generation, evaluation and validation strategies tailored for sparse contradictions. Using real-life documents, we empirically verify that DocChecker detects intra- and inter-document contradictions with an average F1-score of 0.900, up to 0.935, and it scales to large collections $\mathcal{D}$ in parallel.

CCS Concepts: • **Computing methodologies** → **Information extraction**.

Additional Key Words and Phrases: Contradiction detection, Graph rules, Rule discovery, Property graphs, Entity resolution, Natural language inference, Large language models

ACM Reference Format:

Shenglin Chen, Wenfei Fan, Ruochun Jin, and Kexin Ma. 2026. Detecting Contradictions Within and Across Documents. *Proc. ACM Manag. Data* 4, 4 (SIGMOD), Article 267 (September 2026), 30 pages. <https://doi.org/10.1145/3837105>

1 Introduction

A contradiction is a condition in which an entity is simultaneously assigned two or more properties or relationships that lead to a logical inconsistency, *i.e.*, they are logically or semantically incompatible at the same time, such that their joint validity violates domain constraints or undermines internal consistency [18, 79, 112].

Documents contain various contradictions. Examples include: (a) event/entity inconsistencies, *e.g.*, “All donors are between 20–45 years old” vs. “Lisa, who donates her kidney, is 70”; (b) conflicting opinions, *e.g.*, “The project is a breakthrough” vs. “The project had no impact”; and

*Corresponding Author

Authors' Contact Information: Shenglin Chen, College of Computer Science and Technology, National University of Defense Technology, Changsha, Hunan, China, chenshenglin@nudt.edu.cn; Wenfei Fan, University of Edinburgh, Edinburgh, United Kingdom and Shenzhen Institute of Computing Sciences, Shenzhen, China and Beihang University, Beijing, China, wenfei@inf.ed.ac.uk; Ruochun Jin, College of Computer Science and Technology, National University of Defense Technology, Changsha, China, jinrc@nudt.edu.cn; Kexin Ma, College of Computer Science and Technology, National University of Defense Technology, Changsha, Hunan, China, makexin@nudt.edu.cn.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2836-6573/2026/9-ART267

<https://doi.org/10.1145/3837105>

(c) mutually exclusive relations/knowledge, *e.g.*, “Jane and Tom are a married couple” vs. “Jane is Tom’s sister” (cf. [79]). Such contradictions may occur within a single document (*intra-document*) or across related documents (*inter-document*).

Contradictions in documents are common: 21% of news stories contain contradictions [20], and at least 3.3% of facts in the 2025 Wikipedia snapshot are inconsistent. Furthermore, growing amount of AI-generated content makes this problem worse due to hallucinations: 48%–82% of generated text is affected, and factual consistency can drop to as low as 21% in some datasets [99].

The prevalence of textual inconsistencies highlights the need for their detection, since such contradictions can undermine core tasks including information retrieval, named entity resolution (NER), retrieval-augmented generation (RAG), reading comprehension, medical text analysis, and LLM fine-tuning [112, 123, 129, 132].

Early work casts contradiction detection as sentence-pair classification (contradiction or not) [16, 21, 23, 54, 60, 107, 134] or textual entailment (checking whether a premise contradicts a hypothesis) [21, 92, 107, 134]. However, such NLI-based methods are “rather shallow” [57, 88, 136], as they rely on lexical overlap, negation and antonymy, not semantic reasoning. Moreover, cross-document conflicts require more than entity resolution (ER), often involving temporal or causal reasoning; *e.g.*, “Company X was founded in 1998” contradicts “Company X held a board meeting in 1995”, since establishment must precede meetings, an inference beyond ER. Such conflicts are often missed by NLI models [79]. Even state-of-the-art (SOTA) LLMs are only 51.6% accurate [79] in conflict detection, due to noise and hallucinations [139], as they tend to favor parametric knowledge [17] over identifying conflicting evidence [67]. This limitation persists even with few-shot prompting. Although advanced prompting strategies (*e.g.*, self-refine [89] and chain-of-verification [27]) improve reasoning, LLMs still struggle to maintain consistent attention over long contexts and accurately track logical relationships across multiple text spans [27, 89, 110]. As the input grows longer or integrates multiple sources, the benefit of few-shot guidance gradually weakens, making the model more likely to miss structural contradictions and rely on its internal knowledge rather than the input evidence [27, 110].

Challenges. Detecting textual contradictions is challenging.

(1) Context dependence. Contradictions may hinge on context that shifts within or across documents [18, 79]. While ER aligns entity mentions, contextual inference is necessary to catch the inconsistencies. For instance, “the vaccine is safe for children” and “the vaccine has not been tested on minors” seem consistent. However, they become contradictory given the knowledge that any safety claim must be supported by clinical testing.

(2) Linguistic variation. Contradictions may be masked by paraphrasing, negation, or figurative language [8, 108, 112]. The same claim can be phrased in multiple ways; syntactic differences (“X increases risk” vs. “risk is higher with X”) or pragmatic nuances (“may cause” vs. “is known to cause”) can obscure semantic opposition.

(3) Domain specificity. In specialized domains, *e.g.*, medicine, law, or science, contradictions depend on expert knowledge of terminology, causality and regulatory logic [131]. For example, “aspirin prevents blood clots” and “aspirin increases bleeding risk” are both true, but they reflect opposing medical implications of the same mechanism; one statement highlights the benefit of aspirin, the other the risk.

(4) Temporal dynamics. Information evolves over time, and statements that were once consistent may later contradict due to policy changes, scientific updates, or data revisions [79, 87, 111, 112]. For example, “masks are unnecessary” (early 2020) and “masks are required” (late 2020) differ not by error but by temporal context.

(5) *Scalability*. As documents and corpora grow longer, exhaustive pairwise comparison of statements becomes prohibitively expensive [18, 131]. Practical contradiction detection must therefore scale with large, evolving corpora without sacrificing accuracy.

These challenges raise several questions: (1) How can contradictions of different types be detected in a unified way, capturing contextual cues, linguistic variation, and temporal change? (2) How can external domain knowledge be incorporated to boost accuracy when dealing with specialized terminology and context? (3) How can systems efficiently and consistently detect contradictions both within a document and across multiple documents at scale?

DocChecker. To answer these questions, we develop DocChecker, a system for detecting diverse types of contradictions across a document collection $\mathcal{D}$, integrating contextual reasoning, domain knowledge, and scalable analysis. It offers the following features.

(1) *Rules* (Section 3). Central to DocChecker is a class of Rules for Identifying COntradictions (RICOs) over text and graph data. RICOs encode entities and relations as graph patterns, and integrate logical reasoning with ML/NLP methods to detect contradictions. They (a) combine reasoning and prediction to catch context and temporal change, (b) link graph vertices to document spans and use NLP tools to resolve linguistic variation, and (c) infer conflicts in both attributes and relations, catching diverse textual inconsistencies.

(2) *A graph-based framework* (Section 4). Given a document collection $\mathcal{D}$, DocChecker adopts a graph-based paradigm in five stages: (a) convert each document into a property graph, enriching entities and relations with domain knowledge when available; (b) fuse the graphs by aligning vertices referring to the same entity, producing a unified graph G that brings distant statements into structural correspondence; (c) learn RICOs over entity-centric subgraphs sampled from G , reducing the search space while preserving patterns relevant to target entities [33]; (d) detect contradictions via rule-based parallel reasoning over G ; and (e) trace detected conflicts back to their textual origins. This collapses contradictions scattered across long or multiple documents into aligned conflicts, enabling unified intra- and inter-document detection through multi-hop reasoning.

Crucially, DocChecker does not assume that the constructed graph is initially free of contradictions. While candidate rules are generated from entity-centric subgraphs, their final integration into the system is governed by a downstream verification stage. Specifically, each candidate rule is applied to detect and remove contradictions, and is retained only if the resulting cleaned corpus improves the accuracy of a downstream text classifier. Any erroneous rules that fail to improve classification performance are discarded, ensuring that only reliable rules are retained.

(3) *Scalable algorithms* (Sections 5-6). We develop a suite of graph analytic algorithms underlying DocChecker: (a) a graph join algorithm to fuse extracted graphs into a unified graph G and enrich it with domain knowledge from KG, (b) Learner, an algorithm for learning RICOs from (samples of) G , and (c) contradiction detection algorithms for catching inconsistencies in G [1]. These algorithms are designed to scale to long documents and large collections.

The algorithms improve upon prior work. Taking Learner as an example, it (a) replaces costly levelwise traversal [63], which enumerates preconditions step by step, with an LLM that directly generates candidate rules; (b) restricts rule generation to a contradiction-relevant subset $\mathcal{D}'$ where conflicts occur, avoiding full-dataset scans; (c) replaces classical support [32] with downstream performance metrics that suit sparse contradictions (*i.e.*, rare entity-attribute conflicts that are treated by support-based methods as noise); (d) introduces two thresholds (contradiction coverage and accuracy improvement) to ensure broad coverage of contradictions in $\mathcal{D}$; and (e) aligns entities and relations from $\mathcal{D}'$ with triples in G , using their attributes to generate RICO predicates.

Unlike traditional data cleaning rules (e.g., association rules [4]), which are learned from clean data to capture common patterns, RICOs are designed to characterize contradictions. Since contradictions are inherently sparse, learning from the entire clean corpus would assign them extremely low support, causing traditional mining algorithms to discard them as noise. Therefore, DocChecker generates candidate rules only from the contradiction-relevant subset, where conflict patterns are concentrated, making rule discovery both more effective and efficient. These candidate rules are then validated using downstream classification performance, and only those that consistently improve detection accuracy are retained.

(4) Effectiveness (Section 7). Using real-life documents, we empirically verify DocChecker’s accuracy and scalability: (a) it achieves SOTA results on 15 datasets, including 93.1% F_1 on SNLI [16] and 87.8% accuracy on ContractNLI [70]; (b) it excels at long-range reasoning, with 76.81% (intra-) and 86.96% (inter-document) accuracy, beating GPT-4o (51.89%/65.06%); (c) in document-level self-contradiction detection, it provides verifiable evidence (0.962 F_1), exceeding Claude-3.7 and GPT-5.2; and (d) it scales with document size and entity count in parallel, reducing runtime from 7,410.11s to 974.44s with 16 processors, enabling large-scale deployment.

Novelty. Our key contributions are: (1) RICOs: Rules integrating logical, ML and NLP predicates over text-derived graphs to catch semantic contradictions beyond prior graph rules; (2) DocChecker: A graph-based framework for contextual reasoning across documents via graph construction, enrichment and fusion; (3) Learner: A learning-based rule discovery algorithm that departs from levelwise enumeration; and (4) Detector: A PTIME algorithm for parallel intra- and inter-document contradiction detection.

2 Background and Related Work

This section reviews background and previous work.

Documents. Assume a countably infinite set of symbols W for tokens. A document D is represented as a finite token sequence $\langle w_1, w_2, \dots, w_{|D|} \rangle$, where each token $w_i \in W$ denotes a word, punctuation mark or symbol. We assume the sequence is both syntactically valid and semantically coherent. A *substring* of D is a contiguous sequence of tokens $\langle w_i, \dots, w_j \rangle$ for $1 \leq i \leq j \leq |D|$.

Contradictions. Given documents D and D' , a *contradiction* arises if there exist substrings $s \subseteq D$ and $s' \subseteq D'$ that refer to the same entity or relationship o , but assign mutually exclusive values to at least one attribute or property of o under the domain semantics. If $D = D'$, it is *intra-document*. If $D \neq D'$, it is *inter-document*.

Prior research in the NLP community [79, 105, 112] classifies contradictions into seven distinct types, with examples shown in Table 1: (1) *Factual*: contradictions requiring external world knowledge to verify; (2) *Content* (misinformation, emotion, opinion, semantic, and lexical types): conflicts in the attributes of entities or relationships; (3) *Causal*: contradictions where the result does not align with the cause under a clear causal relationship; (4) *Negation*: explicit negation or antonymy (e.g., use of “not,” “never,” or antonyms); (5) *Relational*: conflicts in entity relationships, such as reversed, missing, or mismatched links; (6) *Numeric*: contradictions involving mismatched or out-of-range numbers; and (7) *Temporal*: contradictions caused by conflicting statements across different time points.

Detection. Given a document collection $\mathcal{D}$, *contradiction detection* identifies all contradictory substring pairs (s, s') where $s \in D$ and $s' \in D'$, for $D, D' \in \mathcal{D}$, and outputs their contradiction types T .

Research on contradiction detection spans social media [75, 76, 78], financial reporting [25] and biomedical literature [6, 7, 103, 105]. Early work focused on sentence-level contradictions framed as text classification or NLI tasks [16, 21, 23, 54]. Later studies extend to event-level contradictions (e.g., conflicting reports placing a summit in Paris vs. Berlin) and inconsistent dialogue responses

Types	Sentence 1	Sentence 2	RICOs
(1) Factual	"The Nobel Prize was indeed established in 1901 to commemorate Alfred Nobel"	"Hinton received the Nobel Prize in 2024, which was established in 1901 to honor Shannon's contributions to the field of information."	Rule φ_1
(2) Content	"William was engaged while reading the paper on computer architecture"	"William found the content of the paper on computer architecture to be extremely boring"	Rule φ_2
(3) Causal	"David's video, which was published on YouTube in 2024, was permanently removed that year due to violence."	"Taylor came across this video published by David while browsing YouTube in 2025."	Rule φ_3
(4) Negation	"On April 20, 2025, the router experienced a brief routing disruption at 2:00 PM"	"The router did not report any routing disruptions throughout April 20, 2025."	Rule φ_4
(5) Relational	"The owner of the brand store, Howard, and his wife Sheila have a harmonious relationship after marriage."	"Howard and his wife Emma have a beloved little daughter named Sheila"	Rule φ_5
(6) Numeric	"Howard's brand store sold 29 pairs of women's high heels on June 15, 2025."	"Howard is concerned about the lack of any sales on June 15, 2025."	Rule φ_6
(7) Temporal	"Around 1860, Nobel and other researches believed that nitroglycerin could not treat heart disease"	"After Nobel's death, further research revealed that nitroglycerin is effective for treating heart disease"	Rule φ_7

Table 1. Examples of Contradictions and Corresponding Detection Rules

(e.g., an agent claiming "I am a strict vegetarian" but later "I love eating steak") [24, 49, 66, 95, 140]. Recent work addresses document-level detection via zero-shot inference (*i.e.*, performing contradiction detection without task-specific training examples), multi-hop reasoning (*i.e.*, the process of combining multiple pieces of information to derive the conclusion), and LLM-based methods (*i.e.*, using pre-trained LLMs to predict without extra training) [79, 92, 107, 112, 134, 135]. Benchmarking across GPT-3.5, GPT-4, PaLM-2, and LLaMA-2 shows GPT-4 leading in accuracy and evidence localization [79]. However, gaps still remain: LLMs lack interpretability, suffer from hallucinations, and often struggle with subjective or long-range contradictions [79, 112].

Unlike prior methods, this work proposes a graph-based paradigm for textual contradiction detection: (1) entities and relationships of interest are extracted into graphs, enabling uniform detection of diverse contradictions over both vertices and edges in long reasoning; (2) graphs are "joined" together and enriched with domain knowledge, capturing domain semantics and uniformly catching both intra- and inter-document contradictions; (3) vertices and edges are bound to substrings and coupled with NLP predicates, handling linguistic variation and ambiguity; (4) logic and ML predicates are integrated to reason about inconsistencies, account for context dependence and temporal dynamics, and provide interpretable logical explanations; (5) the process can be conducted in PTIME.

Inconsistency detection. Inconsistency detection has been extensively studied for relational [5, 10, 13, 22, 34, 35, 39, 42, 56, 90, 114, 121] and graph data [32, 36, 38, 40, 44]. Early efforts focused on consistent query answering and constraint-based repair [10, 22, 34, 35], and later extended to ML-based conflict detection and entity resolution [5, 39, 56, 90, 114]. For graphs, research explored dependencies, association rules, rule discovery and data repair [36, 38, 40, 44].

Data cleaning has also been integrated into ML training pipelines, including interactive tools [45], AutoML systems [47], provenance debugging [48, 128], differentiable preprocessing [58, 80], iterative repair [72, 73], boosting-based [71], fairness-aware [115], weak supervision [101], and self-supervised methods [86]. Recent work unifies cleaning with integration, fairness, and federated learning [52, 68, 115], and assesses its influence on ML performance [81].

DocChecker differs from prior studies as follows. (a) Unlike traditional methods for relational tables or semi-structured graphs, it addresses textual contradictions, resolving conflicts in documents and handling linguistic ambiguity absent in earlier work. (b) It expends graph cleaning rules (GCRs) [32] to support expressive contradiction detection in text-derived graphs (Section 3.2). (c) It uniformly detects intra- and inter-document contradictions by leveraging graph join and fusion to integrate multi-document evidence and trace detected conflicts back to their textual sources. (d) We also develop new rule discovery methods. Since prior discovery methods are ineffective for sparse tasks like contradiction detection, we propose new generation, evaluation

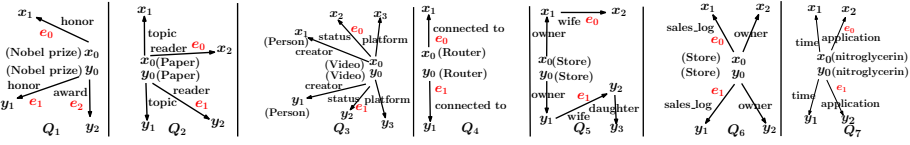


Fig. 1. Example dual patterns

and validation strategies for learning rules effectively in sparse settings (Sections 1 and 6).

3 Contradiction Identifying Rules

This section first extends the notion of dual patterns of [32] (Section 3.1), then defines RICO (Section 3.2), and finally settles the complexity of contradiction detection with RICOs (Section 3.3).

3.1 Dual Graph Patterns

Graphs. A graph is $G = (V, E, L, F_A)$, where (a) V is a finite set of vertices; (b) E is a finite set of labeled edges, with each $e = (v, l, v') \in E$ denoting an edge from v to v' labeled l ; (c) L assigns a label to each vertex; and (d) F_A assigns to each vertex $v \in V$ (resp. edge $e \in E$) a tuple of attributes $(A_1 = a_1, \dots, A_n = a_n)$, written as $v.A_i = a_i$ (resp. $e.A_i = a_i$). Unlike relational data, vertices and edges may carry heterogeneous attributes without a fixed schema.

As opposed to graphs studied in prior work [32, 36, 44] where only vertices are associated with attributes, we consider graphs in which both vertices and edges carry attributes (properties).

Paths. A path ρ from vertex v_0 in a graph $G = (V, E, L)$ is a sequence $(v_0, l_0, v_1, \dots, v_{n-1}, l_{n-1}, v_n)$ such that $(v_{i-1}, l_{i-1}, v_i) \in E$ for all $i \in [1, n]$. We focus on *simple* paths, where no vertex repeats. For an edge $(u, l, v) \in E$, v is a *child* of u , and u is a *parent* of v .

Stars. A star pattern is $Q[x_0, \bar{x}, \bar{e}_x] = (V_Q, E_Q, L_Q)$, where (1) V_Q (reps. E_Q) is the set of pattern vertices (resp. edges); (2) L_Q assigns labels to vertices in V_Q ; (3) $\bar{x}$ and $\bar{e}_x$ are lists of distinct vertex and edge variables, each uniquely denoting an element in V_Q and E_Q ; (4) $x_0 \in \bar{x}$ is the designated *center*; and (5) Q is *star-shaped*: each vertex $x \in \bar{x}$ has a unique path from x_0 and, if $x \neq x_0$, at most one child.

Dual patterns. A dual pattern is $\langle Q_x[x_0, \bar{x}, \bar{e}_x], Q_y[y_0, \bar{y}, \bar{e}_y] \rangle$, denoted $Q[x_0, y_0]$, where Q_x and Q_y are disjoint star patterns. Intuitively, Q couples two entities x_0 and y_0 , each with its own feature leaves, so that their properties can be compared pairwise.

Matches. A match of a star pattern Q in a graph G is a homomorphism h from the pattern vertices and edges in Q to G such that (a) for every $u \in V_Q$, $L_Q(u) = L(h(u))$; and (b) for every edge $(u, l, u') \in E_Q$, $(h(u), l, h(u')) \in E$ holds in G . Similarly, a match of a dual pattern $Q[x_0, y_0] = \langle Q_x, Q_y \rangle$ in G is a homomorphism h from $V_{Q_x} \cup V_{Q_y}$ and $E_{Q_x} \cup E_{Q_y}$ to V and E of G , respectively.

Observe that $h(x_0)$ and $h(y_0)$ may be disconnected in G , which enables comparison of entities without direct links and with differing topological structures in a schemaless graph.

Example 1: Figure 1 depicts seven dual patterns Q_1 – Q_7 , each with two centers (x_0, y_0) and multiple paths. For example, in Q_1 , a path is (y_0, honor, y_1) ; and in Q_5 , a path is $(x_0, \text{owner}, x_1, \text{wife}, x_2)$. $\square$

It is in PTIME to check whether a dual pattern Q has a match in a graph G [32]. In contrast, it is NP-complete to decide whether there exists a match of generic patterns in G (cf. [50]).

3.2 Rules for Identifying Contradictions

We next define RICOs, starting with their predicates.

Predicates. A predicate q of RICOs over a document collection $\mathcal{D}$ and a dual pattern $Q[x_0, y_0] = \langle Q_x[x_0, \bar{x}, \bar{e}_x], Q_y[y_0, \bar{y}, \bar{e}_y] \rangle$ is:

$$q ::= S(x, s) \mid S(e_x, s) \mid M(x.\bar{A}, y.\bar{B}) \mid M(e_x.\bar{A}, e_y.\bar{B}) \mid x.A \oplus y.B \mid z.A \oplus c \mid e_x.A \oplus e_y.B \mid e_z.A \oplus c \mid T(s, s', \tau) \mid \tau(x.A, y.B) \mid \tau(e_x.A, e_y.B).$$

Here s is a *string variable* for a substring in a document; $x \in \bar{x}$ and $y \in \bar{y}$ (resp. $e_x \in \bar{e}_x$ and $e_y \in \bar{e}_y$) are *vertex variables* (resp. *edge variables*) in Q_x and Q_y , respectively; $z \in \bar{x} \cup \bar{y}$ (resp. $e_z \in \bar{e}_x \cup \bar{e}_y$); c is a constant in U ; A and B are attributes in Υ ; τ is a contradiction type such as negation or casual; and $\oplus$ is in $\{=, \neq, <, \leq, >, \geq\}$.

The predicates are classified as follows:

- Substring binding: $S(x, s)$ (resp. $S(e_x, s)$) binds a substring s to a vertex x (resp. edge e_x).
- Textual classifier: $T(s, s', \tau)$ is a NLP/NLI/LLM model that classifies whether substrings s and s' are in contradiction of type τ .
- ML predicate: $\mathcal{M}$ is a pretrained ML model returning a Boolean value (e.g., prediction strength above a threshold), where $x.\bar{A}$ (resp. $e_x.\bar{A}$) denotes a list of attributes on vertex x (resp. edge e_x).
- Comparison: $x.A \oplus y.B$ and $z.A \oplus c$ (resp. $e_x.A \oplus e_y.B$, $e_z.A \oplus c$) compare values of attributes, where $\oplus \in \{=, \neq, <, \leq, >, \geq\}$.
- Type: $\tau(x.A, y.B)$ is true if and only if (iff) $x.A$ and $y.B$ are in contradiction of type τ (similarly for $\tau(e_x.A, e_y.B)$).

Textual classifiers $T(s, s', \tau)$ are pretrained NLP/NLI/LLM models, e.g., (1) SBERT [102] detects attribute-level and negation/antonym contradictions; (2) NumNet [100] handles numeric reasoning; (3) knowledge_graph_maker [31] extracts relations for relational contradictions; (4) Flan-T5-base [19] captures temporal dynamics; (5) GPT-5.2 with retrieval [96] addresses factual contradictions; and (6) Prompted GPT-5.2 [96] detects causal and broad content, misinformation, emotion, opinion, semantic and lexical inconsistencies.

ML predicates $\mathcal{M}(x.\bar{A}, y.\bar{B})$ and $\mathcal{M}(e_x.\bar{A}, e_y.\bar{B})$ can be implemented as pretrained graph-based models, including (1) graph-based NLI classifiers (e.g., GraphSAGE [53], GAT [120]) that embed structural information to detect entailment, neutrality, and contradiction between vertices and edges, (2) graph semantic models (e.g., GraphBERT [138], GCN [69]) that capture event similarity and contextual dependencies among vertices and edges, and (3) graph-based prompt-driven models (e.g., HGT [62]) that extend transformers to infer temporal and causal relationships in graphs.

Rules. A Rule for Identifying COntradictions (RICO) φ has the form:

$$Q[x_0, y_0](X \rightarrow q_0),$$

where $Q[x_0, y_0] = \langle Q_x[x_0, \bar{x}, \bar{e}_x], Q_y[y_0, \bar{y}, \bar{e}_y] \rangle$ is a dual pattern, X is a conjunction of predicates and q_0 is a predicate, both over $\mathcal{D}$ and $\mathcal{Q}$. Moreover, (1) each binary predicate in X is defined on either two leaves x and y in Q_x and Q_y or on their incident edges e_x and e_y , and (2) each leaf and its edge carry at most one binary predicate.

We refer to Q and $X \rightarrow q_0$ as the *pattern* and *dependency* of φ , and to X and q_0 as *precondition* and *consequence* of φ , respectively.

Intuitively, restrictions (1) and (2) above balance expressivity with efficiency following [32] (see Section 3.3). We separate Q and X to (a) visualize the topological features of x_0 and y_0 via Q , and (b) accelerate evaluation by exploiting the locality of pattern matching.

To see the expressivity of RICOs, we compare them with graph cleaning rules (GCRs) [32]. While GCRs are restricted to vertex-level equality and basic similarity, RICOs introduce (1) NLP classifiers for semantic conflicts, (2) edge-level reasoning, (3) structural inference for contextual dependencies, and (4) rich comparison logic (e.g., inequalities) to catch complex numeric and temporal contradictions (see Appendix A in [1] for comparison with prior rules).

Example 2: We illustrate how learned RICOs detect the seven contradiction types in Table 1, using the dual patterns in Figure 1.

(1) *Factual contradiction.* RICO $\varphi_1 : Q_1[x_0, y_0](x_0.\text{title} = \text{Nobel Prize} \wedge y_0.\text{title} = \text{Nobel Prize} \wedge$

$\mathcal{M}(x_0.\text{description}, y_0.\text{description}) \wedge y_2.\text{name} = \text{Hinton} \wedge e_2.\text{time} = 2024 \wedge e_2.\text{award field} = \text{physics} \wedge x_1.\text{name} = \text{Nobel} \wedge y_1.\text{name} = \text{Shannon} \wedge S(e_0, s) \wedge S(e_1, s') \wedge T(s, s', \text{factual}) \rightarrow y_1.\text{name} = \text{Nobel}$). It identifies the factual contradiction in case (1) of Table 1. The contradiction arises because the two awards have nearly identical descriptions (detected by the NLI model $\mathcal{M}(x_0.\text{description}, y_0.\text{description})$) but list different honorees ($x_1.\text{name} = \text{Nobel}$ vs. $y_1.\text{name} = \text{Shannon}$). Using external knowledge and LLM ($T(s, s', \text{factual})$), the rule infers $y_1.\text{name} = \text{Nobel}$. GCRs fail as they (a) lack substring binding ($S(e, s)$) for text localization; (b) skip edge attributes (e.g., $e_2.\text{time}$); and (c) restrict comparisons to equality without NLI models (T).

As noted in Section 4, DocChecker's chase process captures textual bindings and logical conflicts (e.g., $x_1.\text{name} \neq y_1.\text{name}$) via ϕ_1 . Exporting its consequence as a truth value provides users with explicit, verifiable evidence for inspection. This evidence consists of the matched supporting facts (the match of NLI model) and the rule application trace, which together explain why the claims conflict.

(2) *Content contradiction*. RICO ϕ_2 : $\mathcal{Q}_2 [x_0, y_0] (S(e_0, s) \wedge S(e_1, s') \wedge \mathcal{M}(x_1.\text{field}, y_1.\text{field}) \wedge x_2.\text{name} = \text{William} \wedge y_2.\text{name} = \text{William} \wedge e_0.\text{rating} \neq e_1.\text{rating} \rightarrow \text{emotion}(e_0.\text{rating}, e_1.\text{rating}))$. This rule finds an emotion contradiction on edge attributes (case (2) of Table 1), where the same reader ($x_2.\text{name} = \text{William} \wedge y_2.\text{name} = \text{William}$) has conflicting comments (engaged versus boring) on the same paper (identified by $\mathcal{M}(x_1.\text{field}, y_1.\text{field})$). GCRs fail here since they lack edge attributes, inequality predicates ($\neq$), and the text binding ($S(e, s)$) to capture content conflicts.

(3) *Causal contradiction*. RICO ϕ_3 : $\mathcal{Q}_3 [x_0, y_0] (S(x_2, s) \wedge S(y_2, s') \wedge x_1.\text{name} = \text{David} \wedge y_1.\text{name} = \text{David} \wedge x_3.\text{name} = y_3.\text{name} \wedge x_2.\text{details} = \text{removed} \wedge e_0.\text{removed_time} \leq e_1.\text{viewing_time} \wedge y_2.\text{details} = \text{viewable} \rightarrow \text{causal}(x_2.\text{details}, y_2.\text{details}))$. The rule catches a causal contradiction (case (3) of Table 1), as a video cannot be accessible ($y_2.\text{details} = \text{viewable}$) once it has been removed ($x_2.\text{details} = \text{removed} \wedge e_0.\text{removed_time} \leq e_1.\text{viewing_time}$). GCRs fail here because they lack temporal inequalities ($\leq$), ignore edge timestamps (e_0, e_1), and restrict consequences to equality, which preclude semantic predicates such as causal.

(4) *Negation contradiction*. RICO ϕ_4 : $\mathcal{Q}_4 [x_0, y_0] (S(e_0, s) \wedge S(e_1, s') \wedge x_1.\text{ip} = y_1.\text{ip} \wedge e_0.\text{date} = e_1.\text{date} \wedge e_0.\text{interrupt} = \text{true} \wedge e_1.\text{interrupt} = \text{false} \rightarrow \text{negation}(e_0.\text{interrupt}, e_1.\text{interrupt}))$. This rule catches a negation contradiction on edge properties (case (4) of Table 1). Here $\mathcal{Q}_x$ identifies that the router reported an interruption ($e_0.\text{interrupt} = \text{true}$) at 2:00 PM on April 20, 2025, whereas $\mathcal{Q}_y$ indicates no interruption ($e_1.\text{interrupt} = \text{false}$) on the same day. GCRs fail to capture this as they lack negation predicate in consequences, edge attribute comparison that aligns events, and substring binding ($S(e, s)$) which verifies semantic intents like "interruption".

(5) *Relational contradiction*. RICO ϕ_5 : $\mathcal{Q}_5 [x_0, y_0] (S(e_0, s) \wedge S(e_1, s') \wedge x_1.\text{name} = y_1.\text{name} \wedge x_2.\text{name} \neq y_2.\text{name} \wedge e_0.\text{time} = e_1.\text{time} \wedge T(s, s', \text{relation}) \rightarrow \text{relational}(e_0.\text{spouse}, e_1.\text{spouse}))$. This rule catches a relational contradiction that Howard cannot be married to both Sheila and Emma at the same time (case (5) of Table 1). Note that classifier $T(s, s', \text{relation})$ captures context dependence: the contradiction becomes clear only when the contexts of $\mathcal{Q}_x$ and $\mathcal{Q}_y$ are considered together, showing that Howard cannot have two different wives simultaneously ($e_0.\text{time} = e_1.\text{time}$). GCRs fail here since they lack inequality ($\neq$) between entities and edge-level temporal alignments ($e_0.\text{time} = e_1.\text{time}$) to verify simultaneity.

(6) *Numeric contradiction*. RICO ϕ_6 : $\mathcal{Q}_6 [x_0, y_0] (S(e_0, s) \wedge S(e_1, s') \wedge x_2.\text{name} = y_2.\text{name} \wedge e_0.\text{sold items} = \text{women's high heels} \wedge e_0.\text{sale quantity} = 29 \wedge e_1.\text{sold items} = \text{women's high heels} \wedge e_1.\text{sale quantity} = 0 \wedge e_0.\text{date} = e_1.\text{date} \rightarrow \text{numeric}(e_0.\text{sale_quantity}, e_1.\text{sale_quantity}))$. The rule catches a numerical contradiction on the same item that was sold by a store on the same day: sentence 1 reports 29 women's high heels were sold, whereas sentence 2 claims 0 (case (6) of Table 1). GCRs fail here because they cannot handle with numeric edge

attributes, lack numeric predicates in consequence to express quantitative conflicts.

(7) *Temporal contradiction*. RICO φ_7 : $Q_7[x_0, y_0](S(x_2, s) \wedge S(y_2, s') \wedge \mathcal{M}(x_1.\text{time}, y_1.\text{time}) \wedge x_2.\text{treatment} \neq y_2.\text{treatment} \rightarrow \text{temporal}(x_2.\text{treatment}, y_2.\text{treatment}))$. Here $\mathcal{M}$ is an ML model that determines chronological ordering, returning true if $x_1.\text{time}$ precedes $y_1.\text{time}$, e.g., Flan-T5-base model. In this case, $x_1.\text{time}$ is around 1860, while $y_1.\text{time}$ falls after Nobel's death. The rule captures a temporal contradiction: while sentence 1 states that nitroglycerin could not treat heart disease during Nobel's lifetime, sentence 2 claims it was effective after his death (case (7) of Table 1). GCRs fail here as they lack text binding and temporal predicates. The RICO φ_7 integrates NLP predicates with logical interpretability: LLM-based $\mathcal{M}(x_1.\text{time}, y_1.\text{time})$ infers temporal order, while a temporal predicate detects conflicting treatments ($x_2.\text{treatment} \neq y_2.\text{treatment}$). We also employ differentiable methods [58, 80] to jointly optimize ML models and preprocessing by making preprocessing trainable via gradients, allowing end-to-end learning. $\square$

Semantics. Consider RICO $\varphi = Q[x_0, y_0](X \rightarrow p_0)$. A *valuation* of φ is a match h of $Q[x_0, y_0]$ in graph G , extended with bindings of string variables to substrings in document collection $\mathcal{D}$.

We say that valuation h *satisfies* a predicate q , written $h \models q$, if (a) $q = S(x, s)$: vertex $h(x)$ is bounded to substring $h(s)$ in $\mathcal{D}$ (similarly for $q = S(e_x, s)$); (b) $q = x.A \oplus y.B$: $h(x).A \oplus h(y).B$ holds (similarly for $z.A \oplus c$, $e_x.A \oplus e_y.B$ and $e_z.A \oplus c$); (c) $q = \mathcal{M}(x.\bar{A}, y.\bar{B})$: the ML model $\mathcal{M}$ predicts true on features $h(x).\bar{A}$ and $h(y).\bar{B}$ (similarly for $q = \mathcal{M}(e_x.\bar{A}, e_y.\bar{B})$); (d) $q = T(s, s', \tau)$: the textual classifier T assigns contradiction type τ to substrings $h(s)$ and $h(s')$; (e) $q = \tau(x.A, y.B)$: the contradiction between vertex attributes $h(x).A$ and $h(y).B$ is classified as type τ (similarly for $q = \tau(e_x.A, e_y.B)$).

For precondition X , we write $h \models X$ if valuation h satisfies *all* the predicates in X . We write $h \models X \rightarrow q_0$ if $h \models X$ implies $h \models q_0$. We say that graph G and documents $\mathcal{D}$ *satisfy* a RICO φ , written $(G, \mathcal{D}) \models \varphi$, if for all valuations h of $Q[x_0, y_0]$ over G and $\mathcal{D}$, $h \models X \rightarrow q_0$. Similarly, $(G, \mathcal{D}) \models \Sigma$ if $(G, \mathcal{D}) \models \varphi$ for every $\varphi \in \Sigma$.

3.3 The Complexity of Contradiction Detection

We study the *validation problem*, stated as follows.

- Input: A graph G , a document collection $\mathcal{D}$, and a set Σ of RICOs.
- Question: Does $(G, \mathcal{D}) \models \Sigma$?

This problem studies the complexity of contradiction detection.

The validation problem is coNP-hard for GARs [36] but solvable in PTIME for GCRs [32]. Despite being more expressive than both (by supporting edge predicates, substring bindings, and textual classifiers), RICOs still preserve tractability. This contrasts with general graph rules on arbitrary patterns, which are typically intractable [36]. Thus, RICOs strike a balance between expressiveness and efficiency, enabling scalable contradiction detection.

Theorem 1: *The validation problem is in PTIME for RICOs.* $\square$

Proof sketch: An algorithm for RICO validation proceeds as follows: for each φ , it checks whether there exists no valuation h of Q over $(G, \mathcal{D})$ such that $h \models X$ but $h \not\models q_0$. (1) It first locates matches of leaves satisfying X but violating q_0 . (2) It then deduces matches of other vertices in a bottom-up manner. The procedure runs in PTIME, since evaluating predicates such as $S(x, s)$, $T(s, s', \tau)$, $\mathcal{M}(x.\bar{A}, y.\bar{B})$, $\mathcal{M}(e_x.\bar{A}, e_y.\bar{B})$, $\tau(x.A, y.B)$ and $\tau(e_x.A, e_y.B)$ is polynomial, and star-pattern matching is also in PTIME [32]. $\square$

4 An Overview of the System

As shown in Figure 2, DocChecker takes a document collection $\mathcal{D}$ and an optional target entity set $\mathcal{E}$ as input. It catches intra- and inter-document contradictions in five sequential steps as follows.

(1) *Graph conversion.* For each document D in $\mathcal{D}$, DocChecker converts D into a graph G_D using pretrained ML models and LLM, extracting only vertices and edges for entities in $\mathcal{E}$ to exclude irrelevant objects. It records each entity's position in D so that contradictions in G_D can be mapped back to the text. Logically, it defines a relation $S(x, s)$ linking vertices to substrings; physically, this yields a mapping f_{index} from G_D 's vertices to entity positions in D .

In order to incorporate specialized knowledge in the domain and enrich the converted graph, DocChecker joins each G_D and knowledge graph KG when available. The join is conducted only on entities in $\mathcal{E}$, i.e., it extracts from KG only relevant knowledge of users' interest. This yields an enriched graph, also denoted as G_D .

(2) *Graph fusion.* To synthesize the interplay of multiple statements across sentences, sections and documents, DocChecker fuses all graphs G_D into a single graph G . It is carried out by computing the "join" of G_D 's, such that vertices across different graphs are "connected" in G if they refer to the same real-world entity.

(3) *Rule discovery.* Taking a sample of the fused and enriched graph G , DocChecker learns a set Σ of high-quality RICOs with document-specific patterns and domain-specific context, in parallel.

(4) *Contradiction detection.* DocChecker then applies the set Σ of RICOs to G in parallel, to detect contradictions of vertices and edges in G , intra-document and inter-document in a unified process. It guarantees the parallel scalability [74] to scale with large $\mathcal{D}$.

(5) *Document marking.* Using the mapping f_{index} from graphs to documents, DocChecker maps the detected contradictions back to the documents of $\mathcal{D}$, by marking the corresponding text in position.

Below we outline the process of steps 4 and 5 (see details in [1]). We will present steps 1-2 and 3 in Sections 5 and 6, respectively.

Contradiction detector. Step 4 of DocChecker detects conflicts by chasing graph G with RICOs Σ , called Detector (see [1] for details).

Facts. For each attribute $x.A$ (resp. $e_x.A$), the system maintains relations $[x.A]_{\otimes}$ (resp. $[e_x.A]_{\otimes}$), where $\otimes \in \{=, \neq, <, \leq, >, \geq\}$. Each $[x.A]_{\otimes}$ records the attributes and constants that satisfy the $\otimes$ relation with $x.A$, as well as contradiction-type labels; these are derived from a set Γ containing both the initial ground truth Γ_0 and facts generated during the chase. All $[x.A]_{\otimes}$ (resp. $[e_x.A]_{\otimes}$) are valid, e.g., no conflicting constants $c_1 \neq c_2$ may coexist in $[x.A]_{=}$.

Ground truth. Ground truth Γ_0 is a set of consistent facts used as the baseline for reasoning. It is collected from trusted sources and checked by experts (e.g., DBpedia [11]), and fixes key entities and attributes (e.g., setting $[x.A]_{=}$ to constants). A contradiction is detected only when a RICO finds a conflict that violates the facts.

The chase. For a RICO $\varphi : Q[x_0, y_0](X \rightarrow q_0)$ and valuation h satisfying premise X , if current facts $\mathcal{F}$ validate X , consequence q_0 is added to $[x.A]_{\otimes}$, yielding extended set $\mathcal{F}'$ (i.e., Γ). Formally, a *chase step* of graph G with Σ at a set $\mathcal{F}$ of facts is $\mathcal{F} \Rightarrow_{(\varphi, h)} \mathcal{F}'$, which applies φ under the valuation h to produce the updated fact set $\mathcal{F}'$. That is, the chase expands facts Γ with validated facts in the process. A *chasing sequence* ξ is a sequence $\mathcal{F}_0, \dots, \mathcal{F}_n$, where $\mathcal{F}_0 = \Gamma_0$ and each $\mathcal{F}_{i-1} \Rightarrow_{(\varphi, h)} \mathcal{F}_i$ is valid, i.e., it incurs no conflicts. Chase ξ terminates when no rule can extend $\mathcal{F}_n$. When ξ terminates, we refer to the set $\mathcal{F}_n$ of facts as the result of ξ .

Church-Rosser property. The chase ensures deep contradiction detection. Under correct rules and accurate predicates, only correct facts are deduced. Further, the chase is Church-Rosser, i.e., the fixpoint is unique regardless of the order in which rules are applied [2].

Proposition 2: *The chase over G with Σ is Church-Rosser.* □

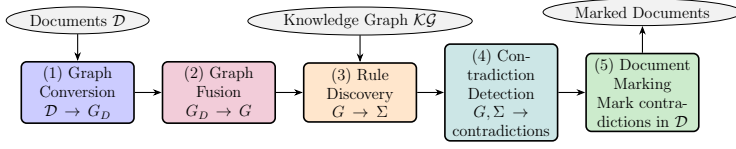


Fig. 2. The workflow of DocChecker

Proof sketch: The proof has two steps. (1) *Finiteness*: Chase steps add facts to a finite number of attributes, bounded by $O(|\Gamma||G| + |G|^2)$. (2) *Convergence*: If two terminal results differ, a missing fix would apply to both, contradicting terminality (see [1]). $\square$

Parallel chasing. Detector implements the chase using the Bulk Synchronous Parallel (BSP) model [119], using one master and K workers. The graph G and document collection $\mathcal{D}$ are partitioned across K workers: (1) each document graph G_D and its document D are colocated on the same worker; (2) all dual-star patterns $Q[x_0, y_0]$ sharing the same x_0 (resp. y_0) and their contexts are placed in the same partition to ensure locality; (3) when G_D or D is large, the associated $Q[x_0, y_0]$ and contexts are further partitioned for load balancing; and (4) each rule φ is dispatched only to the workers containing its corresponding $Q[x_0, y_0]$. Workers execute in parallel, and the master aggregates all newly derived $[x.A]_ =$ (resp. $[e_x.A]_ =$) into Γ .

We show that the algorithm is *parallelly scalable* [74]: its runtime is $t(\mathcal{D}, G, \Sigma)/K$, where $t(\mathcal{D}, G, \Sigma)$ is the cost of the sequential algorithm, and K is the number of processors (see [1]).

Conflict of rules. Although multiple rules may trigger simultaneously on the same entity or relationship, there is no conflict in the deduction, since contradiction types are mutually exclusive (Section 2) and each rule predicts at most one type. Moreover, the reasoning is proved Church-Rosser [1], which guarantees a unique fixpoint that is independent of rule application order.

Incremental detection. In response to updates, Detector employs incremental chase by restricting the rules to an affected area (AFF) determined by updates ΔG . This localized deduction is equivalent to a global computation (see Appendix B.3 in [1] for details).

Document marking. To map graph-level contradictions in G back to their textual origins in $\mathcal{D}$, step 5 of DocChecker maintains, for each triple (h, r, t) in every dual star $Q[x_0, y_0]$, a mapping $f_{\text{index}}(u) : u \mapsto [s_u, e_u]$, where $u \in \{h, t\}$ and $[s_u, e_u]$ is the sentence span in $\mathcal{D}$ containing the entity mention. These spans provide the exact text context for annotating contradictions. If a statement conflicts with multiple others, all involved spans are marked via $f_{\text{index}}(u)$, enabling fine-grained multi-point conflict tracking.

5 Graph Conversion and Fusion

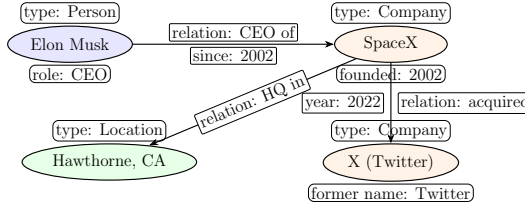
This section presents steps 1-2 of DocChecker: graph conversion (Section 5.1), fusion (Section 5.2) and maintenance (Section 5.3).

5.1 Graph Conversion

Graph conversion transforms each document $D \in \mathcal{D}$ into a graph G_D via LLM-based generation, and enriches the resulting vertex and edge attributes using an external knowledge graph (KG).

LLM-based graph conversion. Following the sampling strategy in [33], DocChecker filters out data irrelevant to entities of $\mathcal{E}$. With a controllable parameter σ , it ensures that RICOs discovered from the samples retain support and confidence with bounded loss (as a function of σ). In other words, it reduces the search space and computational cost while preserving valuable RICOs.

The conversion of D into the property graph G_D follows a three-stage pipeline to ensure semantic consistency. (1) A pretrained model [137] is employed to identify explicit entity mentions and resolves pronouns to antecedent nouns, ensuring that all relations anchor to specific entities.

Fig. 3. Example graph G_D

(2) Cross-document entities are matched, where an LLM (e.g., GPT-5.2 [96]) is used to understand highly varied linguistic expressions, including diverse domain-specific terminologies across multi-genre documents (such as legal contracts, economic reports, and narratives), and cluster similar mentions into canonical entities in $\mathcal{E}$, which prevents redundant nodes. (3) The LLM extracts attributes and relationships from documents, and maps them to node attributes and edge predicates. This translates unstructured text into G_D that preserves logical constraints in documents.

Example 3: Given document D “Elon Musk, the CEO of SpaceX (founded in 2002), acquired X (formerly Twitter) in 2022. SpaceX, a private company, has its headquarters in Hawthorne, CA” and the entity set $\mathcal{E} = \{\text{Elon Musk, SpaceX, X (Twitter), Hawthorne, CA}\}$, the NER model extracts the relevant contexts: (1) Elon Musk, the CEO of SpaceX, (2) SpaceX (founded in 2002), acquired X (formerly Twitter) in 2022, and (3) SpaceX has its headquarters in Hawthorne, CA. These contexts are formatted as a prompt (given in [1]) and used to instruct the LLM, which generates the property graph G_D in Figure 3, capturing the entities and their relationships. $\square$

Graph enrichment. To recover information omitted by LLMs during graph generation [64, 97, 124] and to supplement documents D that are not self-contained in specialized domains, we enrich each generated graph G_D with knowledge from a curated KG (e.g., Wikidata). Given G_D , the KG, and entity set $\mathcal{E}$, we augment G_D with domain-relevant facts about entities in $\mathcal{E}$. Enrichment consists of two steps: (1) entity linking and disambiguation, and (2) selective extraction of KG attributes and relations, combining efficient retrieval with context-based disambiguation and no manual effort.

Entity Linking. We adopt an automatic two-stage strategy to map each entity node $e_i \in \mathcal{E}$ in G_D to its unique identifier q_i in the KG.

(1) *Candidate retrieval.* For each entity e_i , the Wikidata API (wbsearchentities) is queried automatically via standardized templates [122], which supports high throughput [91]. This returns a candidate set $Q_i = \{q_1, \dots, q_k\}$ of the top- k most relevant identifiers ($k = 10$ yields recall of over 90% [30, 59]). Each candidate is attached with its description and aliases from the KG. This step restricts the search space to a compact and accurate set, which provides high scalability to thousands of entities per document [109].

(2) *Contextual disambiguation.* Then, the best match of e_i from Q_i is selected as follows. Taking $C(e_i)$ and $\text{Desc}(q_j)$ as inputs, a SBERT model [102] embeds a context vector $\mathbf{v}_{C(e_i)}$ for e_i and a description vector $\mathbf{v}_{\text{Desc}(q_j)}$ for each candidate q_j in Q_i , where $C(e_i)$ is the KG descriptions for the two-hop neighboring entities of e_i , and $\text{Desc}(q_j)$ is the description of q_j . Then the best matching identifier q^* for e_i is selected with the highest cosine similarity of vectors: $q^* = \arg \max_{q_j \in Q_i} \frac{\mathbf{v}_{C(e_i)} \cdot \mathbf{v}_{\text{Desc}(q_j)}}{\|\mathbf{v}_{C(e_i)}\| \|\mathbf{v}_{\text{Desc}(q_j)}\|}$. Thus, the system resolves ambiguities without manual review. Note that only neighbors within two-hops are considered since they provide enough information for disambiguation with limited cost [30, 51].

Selective extraction. Given G_D with linked KG identifiers, attributes and relations are retrieved via SPARQL to enrich the graph. To ensure relevance, a domain label (e.g., finance, biomedicine) is inferred for each entity using GPT-5.2 [96]. Retrieval is also restricted to a two-hop neighborhood to

be efficient and context-aware, and only domain-specific predicates are retained. Selective extraction preserves essential evidence while reducing noise.

After these two stages, we obtain the enriched graph (also denoted by G_D), which encodes D with augmented KG knowledge.

Example 4: Consider the node John_Smith labeled Person in G_D . The enrichment first links the node to Wikidata entity Q123456 and retrieves attributes occupation : Software_Engineer, citizenship : United_States and educated_at : Stanford_Univ. These attributes are attached to the John_Smith, enabling richer semantic reasoning (e.g., matching by education or nationality). $\square$

Complexity. The graph conversion complexity is $O(NL)$, where N is the number of documents and L aggregates document, prompt, graph lengths, and enriched entities ($L_D, L_P, L_G, |\mathcal{E}|$), parallelizable to $O(NL/K)$ across K processors (refer to Appendix D.1 in [1]).

5.2 Graph Fusion

Graph fusion identifies nodes in different G_D 's that refer to the same real-world entity, links them with a special ϵ -edge, and merges all G_D 's into a unified graph G . Below we first revise the notion of graph join [37] and then extend it to introduce the graph fusion.

Graph join. Graph join identifies entities referring to the same real-world object [37]. Prior methods merge such entities into a single node by unifying their attributes and relations [28], which obscures the provenance of attributes and relations and makes it difficult to trace contradictory facts back to their source documents. Hence we develop a provenance-preserving hybrid alignment method.

Local graph embedding. We use an HGNN-based model (e.g., R-GCN [106]) to learn embeddings $\mathbf{h}_u$ and $\mathbf{h}_v$ for nodes $u \in G_{D_1}$ and $v \in G_{D_2}$, capturing both attributes and local topology. First, SBERT encodes the raw attributes A_u into an initial vector $\mathbf{a}_u$. Then, the HGNN performs structure-aware aggregation: $\mathbf{h}_u = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{v' \in \mathcal{N}_r(u)} \frac{1}{|\mathcal{N}_r(u)|} \mathbf{W}_r \mathbf{h}_{v'} + \mathbf{W}_0 \mathbf{a}_u\right)$, where $\mathcal{R}$ is the set of relation types, and $\mathbf{W}_r$ is the weight matrices.

Matching and ϵ -edges. Given the fused embeddings $\mathbf{h}_u$ and $\mathbf{h}_v$, we compute the similarity between nodes u and v via cosine similarity: $S(u, v) = \cos(\mathbf{h}_u, \mathbf{h}_v)$. If $S(u, v)$ exceeds a threshold τ , then u and v are connected by an ϵ -edge rather than be merged. This preserves the distinct structures and attributes of each entity node in their respective graphs G_D , which enables fine-grained analysis of their interplay during contradiction detection.

The hybrid alignment offers three benefits. (1) SBERT provides semantic similarity even when attribute texts differ literally. (2) HGNN aggregation incorporates 2-hop structural context [30], allowing alignment beyond sparse or inconsistent attributes. (3) The combination is robust to missing or heterogeneous information, linking entities that share limited context or weak structural cues.

Example 5: Consider nodes u and v from documents D_u and D_v that both refer to Elon Musk: (1) u : {name: "Elon Musk", attribute: {"Role": "CEO"}}, (2) v : {name: "Elon Musk", attribute: {"Role": "CTO"}}. Previous graph join [28] merges them into a single node: w : {name: "Elon Musk", attribute: {"Role": "CEO", "CTO"}}. This direct join yields conflicting attributes (e.g., Role: "CEO, CTO"). In contrast, we connect them with an ϵ -edge: (u, v) : {label: "epsilon"}. Our method preserves both nodes and connects them with an ϵ -edge, such that (1) CEO vs. CTO conflict remains explicit for contradiction detection; and (2) nodes u and v stay associated with their documents D_u and D_v , which is essential for document resolution. $\square$

Graph fusion. We merge all G_D 's into a unified graph G , which aligns entity references across documents to provide shared context for matching and conflict detection. Without fusion, pairwise detection on isolated G_D 's may miss cross-document inconsistencies and degrade inter-document

accuracy. Given $\{G_{D_i}\}_{i=1}^N$, we apply a low-complexity composition that links coreferent entities via ϵ -edges while preserving each G_{D_i} . Specifically, (a) for every pair $\langle G_m, G_n \rangle$, we identify coreferences across graph pairs and add ϵ -edges via graph join, and (b) we construct G by uniting all nodes and edges with these ϵ -edges: $G \leftarrow \left(\bigcup_{i=1}^N G_{D_i} \right) \cup \{\epsilon\text{-edges}\}$.

Complexity. The combined graph join and fusion process runs in $O(n \log n + |\mathcal{E}_{\text{total}}|)$, where n is the total number of nodes and $|\mathcal{E}_{\text{total}}|$ the total edges (see Appendix D.2 in [1] for details).

5.3 Incremental Maintenance of Graphs

To handle document updates, DocChecker derives an incremental algorithm $\mathcal{A}'$ from its sequential graph conversion and fusion algorithm $\mathcal{A}$ using localizable computation principles [41, 43]. By restricting execution to the affected area (AFF), $\mathcal{A}'$ ensures that costs scale with the update size rather than the entire corpus.

Proposition 3: *Given $\mathcal{D}$ and an update $\Delta\mathcal{D}$, $\mathcal{A}'$ satisfies $\mathcal{D} \oplus \mathcal{A}'(\mathcal{D}, \Delta\mathcal{D}) = \mathcal{A}(\mathcal{D} \oplus \Delta\mathcal{D})$, where $\oplus$ is the application of updates.* $\square$

This process includes incremental graph conversion, enrichment, and fusion. By localized re-computation within the AFF, the system ensures marginal overhead for dynamic updates. Appendices B.1 and E in [1] provide method details and the proof of Proposition 3.

6 Fast Discovery of RICOs

This section introduces Learner for RICO discovery (Step 3 of DocChecker). We state the discovery problem (Section 6.1), develop Learner (Section 6.2) and present its parallelization (Section 6.3).

6.1 The RICO Discovery Problem

Challenge. RICO discovery poses challenge beyond traditional rule mining [33] where quality metrics, e.g., support, become invalid. As each RICO targets a specific contradiction type and typically involves only a small set of entities, most useful RICOs have low support and would be pruned by conventional high support thresholds.

Evaluation metrics. We replace support with two metrics.

Contradiction coverage: $f_c(C, \Sigma) = \frac{|\{u \in C \mid \text{detected}(u, \Sigma)\}|}{|C|}$ measures the fraction of contradictions correctly detected by rule set Σ . If $\mathcal{D}$ provides fine-grained annotations (location and type), C contains all contradiction instances $\mathcal{I}_C$, and a unit u is detected only if both its location and type are correctly identified. If $\mathcal{D}$ offers only binary labels, C consists of all contradictory segments $\mathcal{S}_C$, and u is detected if Σ correctly marks the segment as contradictory.

Accuracy improvement. We assess rule quality by measuring the improvement in binary classification accuracy (contradictory vs. normal) achieved by the set Σ of RICOs. We employ a basic classifier RoBERTa-large [85] to obtain the binary accuracy $\text{Acc}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} \mathbb{I}(\hat{y}_s = y_s)$, where $\hat{y}_s$ and y_s are the predicted and ground-truth labels, and $\mathbb{I}(\cdot)$ is the indicator function. The improvement is quantified as $f_a(\mathcal{D}_{\text{filter}}, \mathcal{D}) = \text{Acc}(\mathcal{D}_{\text{filter}}) - \text{Acc}(\mathcal{D})$, where $\mathcal{D}_{\text{filter}} = \mathcal{D} \setminus \{s \in \mathcal{D} \mid \Sigma \text{ detects } s \text{ contradictory}\}$. Thus, f_a evaluates whether contradiction filtering effectively enhances the classifier's ability to distinguish normal texts from contradictory ones.

Problem statement. Given a document set $\mathcal{D}$ and its converted, enriched, and fused graph G , the RICO discovery problem is to find a rule set Σ from G such that Σ achieves contradiction coverage at least α and accuracy improvement at least β for bounds α and β .

Algorithm 1: Algorithm Learner

Input: Graph G , corpus $\mathcal{D}$, threshold α , objectives $f_c(\cdot)$ and $f_a(\cdot)$.
Output: Rule set Σ

- 1 Extract generalized dual-star patterns $\mathbb{Q} = \{Q_1, \dots, Q_n\}$;
- 2 Prune $\mathbb{Q}$ via LLM with context $\mathcal{D}_i$; obtain $\mathbb{Q}_s = \{Q'_1, \dots, Q'_s\}$;
- 3 **for** $Q'_i \in \mathbb{Q}_s$ **do**
- 4 Annotate Q'_i with confidence score;
- 5 **if** $score < 0.5$ **then** Reflector LLM decides whether to keep Q'_i ;
- 6 Generate predicate set $\mathcal{P} = \{P_1, \dots, P_s\}$ for $\mathbb{Q}_s$;
- 7 Extract gold standard C from $\mathcal{D}$;
- 8 $\Sigma \leftarrow \emptyset$; $\Sigma' \leftarrow \emptyset$; $iter \leftarrow 0$; $FailCount \leftarrow 0$; $k \leftarrow 3$;
- 9 **while** $f_c(\Sigma, C) \leq \alpha$ **and** $FailCount < k$ **do**
- 10 $\Sigma_{new} \leftarrow \emptyset$;
- 11 **for** $Q'_i \in \mathbb{Q}_s$ **do**
- 12 Generate rules Σ_i via LLM from $\mathcal{P}$;
- 13 Retain $\Sigma'_i \subseteq \Sigma_i$ hitting gold standard C ;
- 14 $\Sigma_{new} \leftarrow \Sigma_{new} \cup \Sigma'_i$;
- 15 $\Sigma' \leftarrow \Sigma' \cup \Sigma_{new}$;
- 16 Generalize predicates in Σ' (e.g., $Smith.age \rightarrow Person.age$);
- 17 Use generalized Σ' as few-shot prompt to guide next iteration;
- 18 **if** $(iter + 1) \bmod k = 0$ **then**
- 19 Apply Σ' to $\mathcal{D}$; obtain $\mathcal{D}_{clean}$;
- 20 **if** $f_a(\mathcal{D}_{clean}) > f_a(\mathcal{D})$ **then**
- 21 $\Sigma \leftarrow \Sigma \cup \Sigma'$; $FailCount \leftarrow 0$;
- 22 **else**
- 23 $FailCount \leftarrow FailCount + 1$;
- 24 $\Sigma' \leftarrow \emptyset$;
- 25 $iter \leftarrow iter + 1$;
- 26 **return** Σ ;

6.2 Discovering RICOs by Learning

Our learning-based discovery algorithm, Learner (Algorithm 1), formulates RICO discovery as a learning-based optimization problem to improve accuracy while avoiding exponential search. By leveraging template-based prompt automation [126] and LLMs, Learner automatically generates rules that satisfy the metrics in Section 6.1.

Learner consists of the following three key modules.

Dual-star pattern extraction and pruning. This module extracts common dual-star patterns from the document graph G and employs an LLM to prune unpromising candidates. It first extracts generalized dual-star patterns $\mathbb{Q} = \{Q_1, Q_2, \dots, Q_n\}$ from G (line 1) via BFS starting from entity nodes connected by ϵ -edges (labeled `same_entity`) up to depth l , e.g., 5 [130]. Each pattern stores only node and edge labels and structure, without attributes such as entity names or values, providing sufficient structural context for rule matching [61] while enabling generalization across entities.

Learner refines $\mathbb{Q}$ using a two-stage pruning pipeline [29] (line 2): (1) an LLM evaluates each $Q_i \in \mathbb{Q}$ against its document context $\mathcal{D}_i$ across seven contradiction types (see [1]). (2) Low-confidence cases (< 0.5) are then routed to a Reflector LLM [110] (lines 3–5) for chain-of-thought cross-validation. Only verified contradiction patterns are retained in the final set $\mathbb{Q}_s = \{Q'_1, Q'_2, \dots, Q'_s\}$.

Predicate generation and gold standard construction. This module generalizes attributes into typed

predicates and builds the gold standard for evaluation. Given $\mathbb{Q}_s$, Learner uses an LLM to generate a predicate set $\mathcal{P} = \{P_1, \dots, P_s\}$ (line 6), formalizing attributes and relations into typed predicates (e.g., comparison predicates $z.A \oplus c$ and textual classifiers $T(s_1, s_2, \tau)$) that compose rule preconditions and consequences. For evaluation, Learner derives a gold standard set C from ground-truth annotations (line 7), including binary labels, contradiction types, and evidence spans [79]. Intuitively, C is a set of constraints specifying which entity-attribute (or relation) assertions cannot co-exist, serving as the objective reference for validating rule accuracy and coverage (see [1] for details).

Three-stage discovery. Given the predicate set $\mathcal{P}$ and gold standard C , Learner initializes two empty sets, Σ' (candidate rules) and Σ (final rules) (line 8), and then iteratively performs three steps.

Stage 1: LLM-based rule generation (lines 11–15). For each candidate pattern $Q'_i \in \mathbb{Q}_s$, the LLM generates an initial rule set Σ_i , using predicates in $\mathcal{P}$ to form preconditions and consequences. Rules contain at most n precondition predicates for a bound n and a single consequence [93, 97], balancing expressiveness and efficiency. Rules are applied to Q'_i , and only those that correctly match the gold standard C are retained as Σ'_i , which are accumulated into Σ' .

Stage 2: Meta-learning feedback (lines 16–17). To enhance generalization, Σ' is used as few-shot guidance. Before prompting, entity- and attribute-specific names are replaced with typed labels (e.g., $\text{Smith.age} \mapsto \text{Person.age}$). The LLM then derives generalized rule patterns, improving subsequent generations.

Stage 3: Verification (lines 18–25). Rules in Σ' are evaluated every k iterations, where k is configurable parameter. Following prior work [89] and our experiments, we set $k = 3$ as LLM outputs typically stabilize within 2–4 iterations, balancing generation quality and computational cost; k can be adjusted in practice. During verification, detected contradictions are mapped back to get $\mathcal{D}_{\text{filter}}$. Rules with $f_a(\mathcal{D}_{\text{filter}}) > f_a(\mathcal{D})$ are added to Σ and the failure counter resets; otherwise, they are discarded and the counter increments.

The process terminates when (a) Σ achieves contradiction coverage $f_c(\Sigma, C) \geq \alpha$, and (b) accuracy improvement $f_a(\mathcal{D}_{\text{filter}}, \mathcal{D}) \geq \beta$.

Incremental RICO discovery. When data is updated, Learner incrementally maintains the rule set Σ by discovering from the affected area (AFF) induced by ΔG . It updates quality metrics (f_c, f_a) via localized statistics in $O(|\Delta G|)$ and restricts LLM-driven discovery to additions, deletions and updates of entities/edges. This incremental discovery bounds the maintenance cost to the update size, which yields high performance (see Appendix B.2 in [1] for details).

Example 6: To illustrate Learner, consider a medical document $\mathcal{D}$ describing patient **John Doe**, where the admission record M_1 reports age 45 and the discharge summary M_2 reports age 65.

Phase 1. Learner identifies a dual-star centered on M_1 and M_2 , linked via ϵ -edges to *John_Doe: Record*(M_1) $\xrightarrow{\epsilon}$ *Person*(P) $\xleftarrow{\epsilon}$ *Record*(M_2). From the associated text, the LLM detects a *Numerical Contradiction* and adds the pattern to $\mathbb{Q}_s$ with confidence 0.95.

Phase 2. From this contradiction, Learner generates typed predicates ($P_{\text{age}}, P_{\text{status}}$, and comparison $z.A \oplus c$) and instantiates textual classifiers for “45-year-old” vs. “age is listed as 65”. These are aligned with the corpus annotations to form the gold standard C .

Phase 3. (1) *Generation*: The LLM constructs a rule such as $M_1.\text{age} \neq M_2.\text{age} \rightarrow \tau_{\text{num}}$, which is added to Σ' after matching C . (2) *Meta-learning*: Specific values are generalized (e.g., $c_1 \neq c_2$ to $x.A \neq y.B$, for unequal attributes across records of the same entity). (3) *Verification*: After periodic evaluation, filtering contradictions improves accuracy on $\mathcal{D}_{\text{filter}}$, and the rule is accepted into Σ . $\square$

Dataset	Data size	Total	Factual	Content	Causal	Negation	Relational	Numeric	Temporal	Vertex	Edge	Acc	P	R	F_1	EHR	Avg. LLM cost
binary-FEVER-NLI	222 docs	105	58	3	1	0	30	4	9	65	40	0.789	0.895	0.869	0.882	0.853	7.2k tokens
CaseHOLD	530 docs	286	78	60	135	1	1	4	7	142	144	0.868	0.879	0.986	0.929	0.864	5.1k tokens
ConTRoL	417 docs	229	57	19	119	1	3	18	12	94	135	0.793	0.895	0.875	0.885	0.854	16.6k tokens
ContraDOC	891 docs	517	146	41	218	5	1	53	53	240	277	0.841	0.885	0.944	0.914	0.860	16.4k tokens
ContractNLI	104 docs	65	3	38	21	1	1	0	1	41	24	0.878	0.878	1.000	0.935	0.865	55.2k tokens
DOCNLI	471 docs	223	38	28	111	0	2	28	16	94	129	0.850	0.883	0.958	0.919	0.861	18.1k tokens
ECON	120 docs	363	239	13	17	9	0	71	14	323	40	0.801	0.893	0.886	0.890	0.855	21.3k tokens
Knowledge Conflict	200 docs	90	63	13	1	0	0	8	5	84	6	0.670	0.918	0.713	0.803	0.838	5.2k tokens
binary-MCTest-NLI	400 docs	217	48	36	114	3	1	5	10	89	128	0.872	0.879	0.991	0.932	0.864	14.7k tokens
MNLI	825 docs	1,309	456	168	419	101	7	85	73	709	600	0.770	0.899	0.844	0.870	0.833	18.2k tokens
Negation	899 docs	397	316	0	15	31	2	22	11	338	59	0.790	0.895	0.870	0.883	0.853	6.4k tokens
Real-life	131 docs	131	68	4	20	2	0	22	15	94	37	0.804	0.892	0.891	0.892	0.855	6.1k tokens
RTE-3	744 docs	549	245	73	120	1	9	69	32	387	162	0.836	0.886	0.937	0.911	0.817	22.5k tokens
SNLI	571 docs	913	391	269	90	12	64	29	58	689	224	0.870	0.879	0.989	0.931	0.813	17.2k tokens
StorySumm	580 docs	81	42	7	25	4	0	2	5	47	34	0.852	0.883	0.961	0.920	0.862	8.1k tokens

Table 2. Accuracy of contradiction detection and evidence retrieval across all datasets

6.3 Learning RICOs in Parallel

The parallel version P_Learner of Learner accelerates the computationally intensive components of the discovery loop, while preserving sequentially dependent stages such as meta-learning feedback. As detailed in [1], P_Learner introduces two modules: parallel rule generation and filtering, and parallel predicate generalization.

Parallel rule generation and filtering. This is the primary source of speedup. P_Learner distributes the candidate dual-star patterns $\mathbb{Q}_s$ across K workers (lines 5–8). Each worker processes its assigned Q'_i independently, generating rules via the LLM and filtering them using the gold standard. The retained rules Σ'_i are aggregated atomically into Σ_{new} , eliminating the latency of sequential LLM calls.

Parallel predicate generalization. This step (e.g., replacing entity names with type labels) is done independently for each rule in Σ' , needs no cross-worker coordination, and is parallelized (line 10).

We have the following (see Appendix H in [1] for a proof).

Proposition 4: P_Learner scales *parallelly* relative to Learner. □

Remark. Learner focuses on contradiction detection (conflicting attributes or relations for the same entity); the identified conflicts can be repaired by existing RICO-based, knowledge-based, or LLM-based methods. Thus, Learner complements existing data repair frameworks and enables full KG cleaning when coupled with a repair module. Note that LLMs are confined to the offline rule discovery and graph conversion phase, where pruning reduces LLM calls. In contrast, RICO execution is online and employs only lightweight ML models (e.g., GCN and SBERT; Section 3.2), and thus enables efficient inference without invoking LLMs.

7 Experimental Study

Using real-life documents, we empirically verify DocChecker’s accuracy, scalability and its performance gains on LLM tasks.

Experimental setting. We start with the setting.

Datasets. We used 15 datasets (sizes in Table 2) for seven tasks: (1) detection of all seven contradiction types; specifically, ContraDoc [79], ECON [65], Knowledge Conflict [125], StorySumm [113], DOCNLI [134], ContractNLI [70], ConTRoL [83] and CaseHOLD [141] support inter-document tasks, and the others focus on intra-document conflicts; (2) sensitivity analysis on ContraDoc; (3) document-level self-contradiction detection (DSCD) for binary decision, ranking and retrieval on ContraDoc and ECON; (4) natural language inference (NLI) at sentence and document levels on CaseHOLD and ContractNLI; (5) domain-specific analysis of legal and economic conflicts on ECON and CaseHOLD; (6) scalability analysis of Learner, Detector and DocChecker on six representative datasets; and (7) ablation study on StorySumm.

These datasets span diverse domains, including legal contracts (ContractNLI, CaseHOLD), economic reports (ECON), news (ContraDoc, DOCNLI), and narratives (StorySumm), covering a wide range of real-world genres and writing styles.

Baselines. We compared DocChecker with 15 recent baselines. (1) *Document-level contradiction tasks*. (a) Sensitivity analysis tasks: GPT-4o [3]. (b) DSCD tasks: GPT-5.2-chat-latest [96], Claude-3-7-sonnet-latest [9], Gemini-3-pro-preview-thinking [116], Llama-3-70B [118], Qwen-Max-Latest [12], DeepSeek-V3 [82], Longformer-base-4096 [14] and RoBERTa-large-MNLI [85]. (2) *NLI tasks*. (a) Document-level: Longformer-base-4096, BART-base [77], DeBERTa-v3-base [55], DistilBERT-base [104] and RoBERTa-large-MNLI. (b) Sentence-level: BERT-base-uncased [26], RoBERTa-base [85], XLNet-base-cased [133] and RoBERTa-large-MNLI.

Prompts. To assess prompts on LLM baselines, we used six prompts with different strategies and tested their sensitivity (Exp-1), DSCD tasks (Exp-1), and a case study (Exp-4) (see Appendices G.3-G.5 [1]).

Ground truth. All datasets provide expert-annotated ground truth labels, including correct vs. distractor answers in QA datasets (e.g., CaseHOLD, ConTRoL), three-way labels in NLI datasets (e.g., SNLI, MNLI, ContractNLI), and contradiction-type labels in others (e.g., binary-FEVER-NLI). See Appendices I-J in [1] for details.

Metrics. We evaluated contradiction detection, classification and evidence retrieval using: (1) *Contradiction metrics* [15, 16]: (a) Accuracy ($\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$) to quantify the proportion of correct predictions, as a standardized metric for benchmarking our approach against NLP/ML baselines [14, 46, 55, 77, 85], where TP, TN, FP and FN denote true positives, true negatives, false positives, and false negatives, respectively; (b) Precision ($P = \frac{\text{TP}}{\text{TP} + \text{FP}}$), the ratio of true positives to all predicted positives; Recall ($R = \frac{\text{TP}}{\text{TP} + \text{FN}}$), the ratio of true positives to all actual positives; and F_1 ($= \frac{2PR}{P+R}$), the harmonic mean of precision and recall. These provide a standard and comparable basis for benchmarking against NLP/ML baselines. (2) *Evidence Hit Rate (EHR)* [117]: the probability $P(V(E) = 1 \mid J(D) = 1)$, where $J(D)$ indicates correct contradiction prediction and $V(E)$ correct evidence localization, measuring whether the model can localize contradictions once detected. (3) *Match Rate (MR)* [100]: the fraction of gold evidence tokens covered by retrieved snippets, quantifying semantic/token-level overlap with ground truth.

Data partitioning and model training. For pre-partitioned datasets, we followed the original splits. For others, we randomly divided data into training/validation/test sets with a 7:1:2 ratio. Baselines were trained, rule patterns learned, and LLM few-shot prompting performed on the training set. Hyperparameters were tuned on the validation set, and final results are reported on the test set.

Environment. DocChecker is implemented in Python. All experiments were conducted in PyTorch on a single server with four NVIDIA GPUs (32GB each) and an AMD EPYC 9754 128-core CPU. Each GPU node was allocated 24 vCPUs and 72 GB RAM, totaling 96 vCPUs and 288 GB memory. We adopted data-parallel training across all GPUs and report the average of three independent runs.

Experimental findings. We next report our findings.

Exp-1: Accuracy. We evaluated DocChecker’s accuracy and robustness against SOTA LLMs and specialized models across benchmarks: (1) contradiction detection with location, type and count annotations (ContraDoc [79]); (2) sensitivity to contradiction position, type and domain (ContraDoc [79]); (3) DSCD [79, 112]; (4) document-level [65, 70, 134] and sentence-level NLI [16, 127].

(1) Contradiction detection. As shown in Table 2, we evaluated DocChecker on 15 datasets. To analyze structural contradiction patterns, we map the seven contradiction types onto our graph representation as: *Vertex-level (entity) contradictions*: conflicts within a single node, where attributes or facts about the same entity disagree. *Edge-level (relation) contradictions*: conflicts expressed along an edge, where the logical, causal, or directional relation between two entities is inconsistent. Mapping

all contradiction types onto vertices or edges enables quantifying their structural distribution.

(a) *Contradiction type distribution.* Table 2 shows substantial variation in the distribution of the seven contradiction types across datasets. (i) Factual and Causal inconsistencies dominate many datasets (e.g., MNLI: 456 factual, 419 causal; ContraDoc: 218 causal), and DocChecker remains stable on these categories, achieving high F_1 on SNLI (93.1%) and ContraDoc (91.4%). (ii) Relational, Numeric, and Temporal contradictions are sparser but more challenging. For instance, Relational cases appear frequently in SNLI (64) but are nearly absent in ECON, which instead contains many numeric conflicts (71). Despite such diversity, DocChecker attains 80.1% accuracy on ECON, evidencing strong robustness.

(b) *Performance and generalization.* (i) DocChecker generalizes well across domains, achieving high F_1 on sentence-level NLI (SNLI: 93.1%, RTE-3: 91.1%). (ii) It remains robust on document-level datasets (ContraDoc: 91.4%, DOCNLI: 91.9%), showing effective long-range modeling. (iii) Evidence Hit Rate (EHR) is consistently high (average 85.7%), including CaseHOLD (86.4%) and StorySumm (86.2%), confirming accurate graph-based evidence retrieval.

(c) *Sensitivity to graph structure.* DocChecker’s accuracy is influenced by how contradictions manifest structurally. Performance lags on vertex-level conflict datasets (e.g., Knowledge Conflict: 80.3%) than those with more edge-level ones (e.g., ContraDoc: 91.4%). This shows that contradiction localization is highly dataset-dependent and underscores the need for explicit graph structure modeling to capture both attribute-level and relational conflicts.

(d) *LLM cost.* We measured the LLM token consumption of DocChecker in its LLM-driven stages, which include (i) graph conversion, where LLMs structure raw documents as graphs, and (ii) rule discovery, where LLMs are employed in the three-stage rule discovery process. As shown in Table 2, the average consumption is 15.9k tokens per document, ranging from 5.1k (CaseHOLD) to 55.2k (ContractNLI), which shows DocChecker’s cost-effectiveness in targeted LLM application during rule inference and learning.

(2) *Sensitivity analysis.* We examined how contradiction position, type, and domain affect DocChecker and GPT-4o on ContraDoc.

(a) *Contradiction positions.* Table 3 tests performance across three contextual spans: local (intra-paragraph), intra-document (inter-paragraph), and inter-document. (i) DocChecker performs strongly on local contradictions (80.82% Acc, 83.56% MR), beating GPT-4o (68.09% Acc). (ii) Accuracy drops to 76.81% for intra-document cases, reflecting the difficulty of detecting subtle cross-paragraph inconsistencies within shared context. (iii) Accuracy rises to 86.96% for inter-document contradictions, showing that graph alignment effectively captures structural divergences across sources. Across all spans, DocChecker consistently outperforms GPT-4o, whose accuracy degrades sharply on long-range contexts (down to 51.89%).

(b) *Contradiction types.* Table 3 examines detection difficulty across the seven contradiction types. (i) DocChecker performs best on structurally explicit categories, achieving 85.00% (relational) and 83.74% (causal), where RICO rules effectively capture clear logical dependencies. (ii) Performance is lower on semantically subtle or sequential types such as negation (68.00%) and temporal (72.86%), due to reliance on fine-grained semantic scope [94] and temporal reasoning [98], which are harder to generalize from structural patterns alone. (iii) Although GPT-4o shows high accuracy on negation and temporal (> 93%), its extremely low F_1 scores (5.00%, 52.17%) indicate severe false positives, whereas DocChecker remains balanced and reliable ($F_1 \geq 80.95\%$ for all types).

(c) *Domain-specific analysis.* Table 3 shows that DocChecker performs most stably on Stories (84.95% Acc.) and News (84.40%), with a high MR in Stories (95.70%), indicating reliable conflict localization.

Dimension	Category	DocChecker		GPT-4o	
		Acc (%)	MR/F ₁ (%)	Acc (%)	MR/F ₁ (%)
Position (task 1)	Local	80.82	83.56	68.09	21.28
	Intra-document	76.81	89.04	51.89	17.92
	Inter-document	86.96	85.51	65.06	20.48
Type (task 2)	Factual	80.30	89.08	75.50	50.99
	Content	80.22	89.03	84.93	30.53
	Causal	83.74	91.15	67.88	18.49
	Negation	68.00	80.95	93.71	5.00
	Relational	85.00	91.89	91.89	3.92
	Numeric	75.25	85.87	94.04	52.63
Domain (task 3)	Temporal	72.86	84.30	94.54	52.17
	News	84.40	90.83	62.39	20.18
	Stories	84.95	95.70	72.04	30.11
	Encyclopedia	74.47	80.85	56.38	22.34

Table 3. Sensitivity analysis of DocChecker vs. LLMs

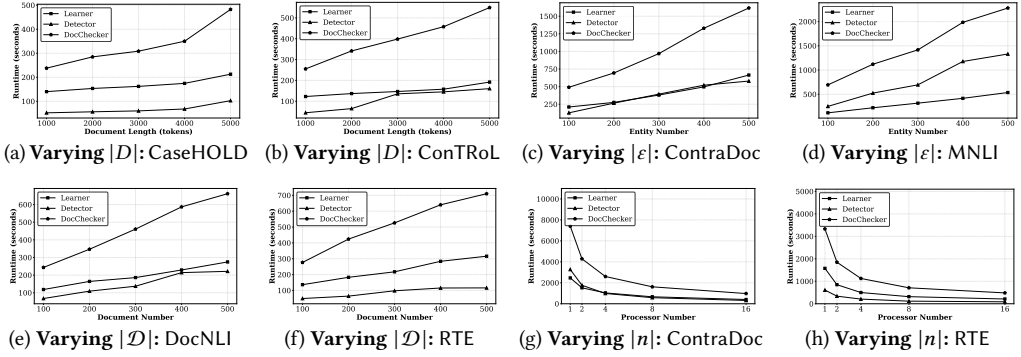


Fig. 4. Scalability trends across key variables

Performance is relatively lower on Encyclopedia text (74.47%) due to dense factual cross-referencing. This said, DocChecker remains consistently more robust than LLM baselines across all domains.

(3) *DSCD tasks*. We tested DocChecker on three DSCD subtasks.

(a) *Binary classification*. This task evaluates whether a method is able to distinguish contradictory (positive) from non-contradictory (negative) texts. As shown in Task 1 of Table 4, DocChecker achieves near-perfect accuracy, reaching $P = 1.000$ and $F_1 = 0.992$ on ContraDoc, far surpassing GPT-5.2 ($F_1 = 0.727$) and Longformer ($F_1 = 0.520$). On ECON, although the performance gap narrows (e.g., $F_1 = 0.859$ by Claude-3-7), DocChecker remains superior with $F_1 = 0.965$. These results indicate that DocChecker’s graph-based modeling captures long-range dependencies more effectively than Transformer architectures and remains robust across domains.

(b) *Top-3 contradiction evidence*. Evaluated on CONTRADOC-POS, this task measures a model’s ability to localize salient evidence sentences. As shown in Task 2 of Table 4, all models achieve $P = 1.000$ due to the positive-only setting, so Recall and Accuracy indicate evidence retrieval success. DocChecker attains the best results with $F_1 = 0.962$ and $\text{Acc} = 0.927$, showing that graph fusion effectively highlights contradictory nodes. In contrast, GPT-5.2 and Claude-3-7 yield $F_1 = 0.856$ and $F_1 = 0.720$, frequently missing key evidence, while Gemini-3-Pro drops sharply ($F_1 = 0.390$); thus strong binary judgment does not necessarily imply reliable evidence localization.

(c) *Evidence identification after binary judgment*. This task tests whether a method can correctly locate contradictory evidence given a positive prediction. In Task 3 of Table 4, DocChecker achieves $J(D) \wedge V(E) = 0.914$, far surpassing GPT-5.2 (0.543) and Claude-3-7 (0.420). While LLMs maintain moderate $J(D)$ (0.56–0.72), they rarely provide verifiable evidence. In contrast, DocChecker attains high EHR (0.942) with balanced Precision/Recall (0.900/0.928), showing accurate localization. This verifies that explicit graph reasoning enables reliable DSCD with verifiable evidence.

(d) *Efficiency and cost.* As shown in Table 4, DocChecker processes each document in 2.02s with 16.8k tokens for preprocessing and rule discovery, and incurs zero LLM cost at inference. For Task 1 (16.3k tokens), it runs in 1.92s, much faster than Llama-3-70B (5.68s); for Tasks 2 and 3, it incurs minimal overhead (0.13s and 0 tokens), while Gemini-3-Pro takes up to 3.63s and 2.4K additional tokens.

In DocChecker, LLM token costs (Table 4) stem entirely from Task 1’s offline graph conversion and rule discovery. By reusing these artifacts, Tasks 2 and 3 operate as subsequent reasoning steps that incur zero additional LLM overhead. Specifically, Task 2 extracts evidence via chasing and document markings, while Task 3 aligns it with binary judgments through a subgraph consistency check. Unlike LLM baselines, DocChecker eliminates prompting costs in Tasks 2 and 3 by reusing graphs and rules, enabling highly efficient, high-throughput processing of large corpora.

(4) *Document-level NLI.* This task assesses ability to reason over long documents by classifying document pairs into *entailment*, *neutrality*, or *contradiction* (Table 5). DocChecker performs strongly in complex legal contexts, achieving 86.8% Acc and 0.929 F_1 on CaseHOLD, and 87.8% Acc with 0.935 F_1 on ContractNLI. Notably, DocChecker attains perfect Recall (1.000) on ContractNLI and surpasses Longformer by nearly 22% in F_1 . Its consistent advantage over specialized models (e.g., DeBERTa-v3-base, BART-base) highlights the effectiveness of RICO-based reasoning: by explicitly modeling document structure and integrating dispersed evidence through a unified graph, DocChecker captures long-range logical dependencies that standard neural attention often misses.

The results of sentence-level NLI tasks are consistent (see [1]).

Exp-2: Scalability. We assessed the scalability of Learner, Detector, and DocChecker by varying: (1) document length $|D|$, (2) entity count $|\mathcal{E}|$, (3) dataset size $|\mathcal{D}|$, and (4) processor number $|n|$.

(1) *Varying document length $|D|$.* We evaluated scalability when $|D|$ grows from 1k to 5k tokens on CaseHOLD and ConTRoL. Figures 4(a)–(b) show: (a) Learner takes 140.19–212.79s on CaseHOLD and 122.74–191.87s on ConTRoL, driven by the number and complexity of extracted dual-star patterns, which increase with document length. It extracts 487–577 rules (avg. 528) on CaseHOLD and 862–943 (avg. 896) on ConTRoL. (b) Detector remains stable, ranging from 51.98–102.57s on CaseHOLD and 44.87–160.44s on ConTRoL, as reasoning operates on dual-star graphs with direct span mapping via f_{index} . (c) End-to-end DocChecker takes 237.94–482.53s on CaseHOLD and 255.19–549.18s on ConTRoL, covering graph construction, fusion, rule discovery, detection and marking. Even at 5k tokens, graph construction and join account for only 21.04% (101.54s) on CaseHOLD and 16.76% (92.07s) on ConTRoL, indicating that graph abstraction decouples reasoning cost.

(2) *Varying entity number $|\mathcal{E}|$.* Figures 4(c)–(d) report runtime on ContraDoc and MNLI as graph size (driven by $|\mathcal{E}|$) grows from 100 to 500 nodes. (a) Learner increases from 210.74–664.56s on ContraDoc and 117.72–536.85s on MNLI, driven by denser dual-star patterns; LLM pruning remains stable. (b) Detector ranges from 127.86–579.24s on ContraDoc and 250.94–1,331.25s on MNLI. Despite growth, dual-star topology localizes matching, keeping per-contradiction latency within 278.48ms and 594.31ms, respectively. (c) End-to-end DocChecker increases from 491.11–1,616.97s on ContraDoc and 693.93–2,279.40s on MNLI. As 78.70% of real-world dual-star patterns involve < 200 entities, average runtime remains about 692s even at $|\mathcal{E}| = 500$, i.e., no exponential blowup.

(3) *Varying dataset $|\mathcal{D}|$.* Figures 4(e)–(f) evaluate scalability as the corpus grows from 100 to 500 documents on the DocNLI and RTE datasets. (a) Learner increases moderately from 118.34s to 274.69s on DocNLI and 136.21s to 315.94s on RTE, as iterative pruning limits redundant patterns and LLM calls. (b) Detector maintains stable per-document latency (0.443s on DocNLI and 0.232s on RTE), with total runtime bounded to 221.46s (DocNLI) and 116.01s (RTE) on 16 processors, showing reasoning is localized to dual-star subgraphs rather than corpus size. (c) DocChecker rises from 243.59s to 661.61s on DocNLI and 276.63s to 710.23s on RTE, confirming that offline discovery scales with

Task 1: Binary Classification (CONTRADOC / ECON)							
Model	Prec.	Rec.	F1	Acc.	Time (s)	Tokens (k)	
DocChecker	1.000 / 0.960	0.985 / 0.970	0.992 / 0.965	0.986 / 0.953	1.92/2.83	16.3/21.2	
GPT-5.2	0.983 / 0.941	0.577 / 0.480	0.727 / 0.636	0.608 / 0.633	2.92/6.22	15.9/19.3	
Claude-3-7	0.977 / 0.941	0.433 / 0.790	0.600 / 0.859	0.477 / 0.827	2.04/3.11	16.7/23.4	
Llama-3-70B	0.971 / 0.971	0.340 / 0.330	0.504 / 0.493	0.393 / 0.547	5.68/3.77	16.4/20.1	
Gemini-3-Pro	1.000 / 0.930	0.400 / 0.400	0.571 / 0.559	0.640 / 0.580	4.31/3.53	16.3/22.0	
Longformer	0.650 / 0.353	0.433 / 0.060	0.520 / 0.103	0.520 / 0.300	0.29/0.27	-/-	

Task 2: Evidence Detection (CONTRADOC-POS)							
Model	Prec.	Rec.	F1	Acc.	Time (s)	Tokens	
DocChecker	1.000	0.927	0.962	0.927	0.13	0.0	
GPT-5.2	1.000	0.747	0.856	0.747	0.40	1,520.0	
Claude-3-7	1.000	0.562	0.720	0.562	6.91	1,466.4	
Llama-3-70B	1.000	0.412	0.584	0.412	28.09	1,215.4	
Gemini-3-Pro	1.000	0.242	0.390	0.242	3.63	1,283.9	
RoBERTa-mnli	1.000	0.322	0.413	0.322	0.09	-	

Task 3: Joint Evidence Identification (CONTRADOC)							
Model	$J(D)$	$J \wedge V$	EHR	Prec.	Rec.	Time (s)	Tokens
DocChecker	0.992	0.914	0.942	0.900	0.928	0.55	0.0
GPT-5.2	0.719	0.543	0.957	0.537	0.550	0.24	354.1
Claude-3-7	0.682	0.420	0.586	0.415	0.425	0.90	166.8
Qwen-Max	0.561	0.361	0.813	0.406	0.325	2.27	142.8
Gemini-3-Pro	0.625	0.400	0.950	0.346	0.475	11.9	2,379.0
DeepSeek-V3	0.644	0.356	0.684	0.394	0.325	0.62	304.7

Table 4. Performance evaluation of three DSCD subtasks

input volume while online reasoning remains effectively size-invariant for large-scale monitoring.

(4) *Varying processor number $|n|$* . Figures 4(g)–(h) test scalability as n ranges from 1 to 16 on ContraDoc and RTE. (a) Learner scales near-linearly, reducing runtime from 2,465s to 408s on ContraDoc and 1,577s to 212s on RTE, as parallel workers mask LLM latency. (b) Detector maintains stable per-document time (0.4s), while corpus-level runtime drops from 3,270s to 315s on ContraDoc and 609s to 88s on RTE. (c) End-to-end DocChecker drops from 7,410s to 974s on ContraDoc and 3,330s to 484s on RTE, verifying parallel scalability.

Exp-3: Ablation. Table 6 reports the impact of key DocChecker components to effectiveness (Accuracy) and efficiency (Runtime) via five ablations on StorySumm: (a) *Graph join*: Full DocChecker (with cross-document linking) vs. a no-aggregation variant. (b) *Graph enrichment*: Full DocChecker (with external KG enrichment) vs. no-enrichment variant. (c) *Entity resolution (ER)*: Full DocChecker (SBERT + HGCN) vs. no-ER variant (simulating noisy linking). (d) *Meta-learning*: Full DocChecker vs. variants without LLM-based rule generation (using traditional level-wise discovery instead, under a support threshold of 3 and a 5-hour timeout where Learner returns all rules), meta-learning feedback, or verification/filtering.

(1) *Graph components*. Removing graph join causes a substantial drop, reducing accuracy from 85.2% to 63.0%, which confirms cross-document linking as one of the most critical factors for accurate contradiction detection. Disabling Graph Enrichment also degrades accuracy (73.8%), showing that external knowledge provides useful context, although its effect is secondary to structural linking.

(2) *Impact of ER quality*. Inaccurate ER results in false positives as naming ambiguities trigger spurious contradictions. Specifically, disabling SBERT and HGCN in graph fusion increases the FPR of entity-matching by 21.2% and decreases detection accuracy by 16.7% (Table 6). This confirms that accurate entity resolution is vital for ensuring the reliability of downstream contradiction detection.

(3) *Impact of meta learning components*. Table 6 quantifies the impact of three meta-learning components on rule quality and efficiency. (1) Stage 1 (LLM generation): Replacing LLM-driven initialization with a levelwise strategy drops accuracy by 38.0%, proving the LLM’s role in capturing complex semantic preconditions. (2) Stage 2 (meta-feedback): Disabling iterative feedback decreases accuracy and F1 by 16.6% and 19.9%, respectively. Thus, few-shot guidance is essential for pattern refinement. (3) Stage 3 (verification/filtering): Removing this stage may introduce useless rules that

Model	CaseHOLD				ContractNLI			
	Acc	P	R	F1	Acc	P	R	F1
DocChecker	0.868	0.879	0.986	0.929	0.878	0.878	1.000	0.935
Longformer-base-4096	0.310	0.347	0.310	0.321	0.712	0.734	0.712	0.716
DeBERTa-v3-base	0.676	0.677	0.676	0.676	0.704	0.733	0.704	0.702
BART-base	0.672	0.680	0.672	0.673	0.508	0.451	0.508	0.477
DistilBERT-base	0.605	0.608	0.605	0.606	0.684	0.699	0.684	0.680

Table 5. Performance on legal QA and document-level NLI

fail to detect contradictions, reducing accuracy to 54.7%. This proves the efficacy of failure counter in pruning invalid candidates via $f_a(\mathcal{D})$.

(4) *Efficiency and scalability.* Despite added structural modeling, the full model remains efficient (551.3s), beating ablated variants by a large margin. This shows that DocChecker achieves a strong accuracy–efficiency balance and scales well to dense text analysis.

Exp-4: Case study and contradiction analysis. LLMs struggle with long-range dependencies and implicit logic. We present three ContraDoc cases where DocChecker outperforms GPT-5.2.

(1) *Content contradiction (ID: 3488771835_6)*: In the book *Mrs. Frisby*, a character “agrees wholeheartedly” but later “leaves the colony”. LLMs miss the mismatch between intent and action. In contrast, DocChecker applies $\varphi_{\text{persp}} : \mathcal{Q}[x_0, y_0](S(x_1, s) \wedge S(y_1, s') \wedge \mathcal{M}(x_0.\text{name}, y_0.\text{name}) \wedge \mathcal{M}(x_1.\text{goal}, y_1.\text{action}) \wedge T(s, s', \text{opinion}) \rightarrow \text{content}(x_1.\text{goal}, y_1.\text{action}))$, binding goal (x_1) and action (y_1) to the same entity to detect the conflict.

(2) *Temporal contradiction (ID: 3488771840_9)*: In *Mrs. Tittlemouse*, cleaning duration shifts from a “fortnight” to “a day”. LLMs miss such long-range temporal inconsistencies. DocChecker employs $\varphi_{\text{temp}} : \mathcal{Q}[x_0, y_0](\mathcal{M}(x_0.\text{event}, y_0.\text{event}) \wedge x_1.\text{val} \neq y_1.\text{val} \wedge x_1.\text{unit} = y_1.\text{unit} \rightarrow \text{temporal}(x_1.\text{val}, y_1.\text{val}))$, aligning events and comparing values; $x_1.\text{val} \neq y_1.\text{val}$ catches conflicts across >15 sentences.

(3) *Factual contradiction (ID: 3488771846_3)*: In *A Tale of Two Cities*, Darnay is “imprisoned in Paris” yet “dining in London” on the same day. LLMs miss this due to weak spatial consistency. DocChecker uses $\varphi_{\text{space}} : \mathcal{Q}[x_0, y_0](\mathcal{M}(x_1.\text{time}, y_1.\text{time}) \wedge x_2.\text{loc} \neq y_2.\text{loc} \wedge T(s, s', \text{factual}) \rightarrow \text{factual}(x_2.\text{loc}, y_2.\text{loc}))$, aligning the entity on a shared timeline to catch impossible co-occurrence.

As limitations, we note that contradictions requiring purely subjective or ambiguous interpretation remain challenging.

Contradiction analysis. To mitigate attention dilution in LLMs over long sequences [84], DocChecker aligns scattered text spans into a unified graph, enabling variable binding for logical rule evaluation. This allows DocChecker to accurately detect intra- and inter-document contradictions on ContraDoc, where LLMs underperform by 15.2% due to long-range dependency failures. While such long-span conflicts account for 28.4% of ContraDoc cases, DocChecker remains robust, highlighting the need for explicit graph-based reasoning in complex inter-document settings.

Exp-5: Incremental study. DocChecker incrementally processes updates of $\Delta N = 100$ documents across the CaseHold and MNLI datasets (Table 7). (a) Graph conversion: By restricting extraction to modified fragments, conversion time on CaseHold drops from 72.3s to 16.2s (a 77.6% latency reduction), and tokens per document fall from 766.8 to 166.7 (a 78.3% token reduction). (b) Rule generation: Isolating affected subgraphs reduces MNLI LLM costs from 8,809.5 to 402.8 tokens per document, with processing time dropping from 1,135.3s to 50.1s. (c) Meta-learning: The feedback loop refines Σ in just 20.6s on MNLI, requiring only 4.3% of the initial discovery time.

Summary. We find the following. (1) *Effectiveness.* DocChecker achieves SOTA accuracy on 15 datasets, e.g., 93.1% F_1 on SNLI and 91.4% on ContraDoc. It excels at document-level NLI (87.8% Acc on ContractNLI), outperforming DeBERTa-v3. RICOs (φ_{persp} , φ_{temp} , φ_{space}) resolve complex content,

Model Configuration	Accuracy (%)	F1-score (%)	Runtime (s)
DocChecker (Full Model)	85.2	92.0	551.3
Ablations			
w/o Graph Join	63.0	77.3	481.1
w/o Graph Enrichment	73.8	85.0	445.7
w/o entity resolution & linking	68.5	71.6	377.6
w/o LLM-based rule generation	47.2	52.4	over 5 hours
w/o Meta-learning feedback	68.6	72.1	390.3
w/o Rule verification	54.7	59.8	414.6

Table 6. Ablation study of key DocChecker components.

Dataset	Components	Full Re-computation		Incremental Update	
		Time	Avg. LLM Cost	Time	Avg. LLM Cost
CaseHold	Graph Conversion	72.3s	766.8 tokens	16.2s	166.7 tokens
	Rule Generation	312.3s	3,118.3 tokens	25.2s	231.0 tokens
	Meta-learning Feedback	195.4s	1,249.8 tokens	14.4s	96.1 tokens
	Total Overhead	580.0s	5,134.9 tokens	55.8s	493.8 tokens
MNLI	Graph Conversion	216.3s	4,538.7 tokens	52.5s	1,055.6 tokens
	Rule Generation	1,135.3s	8,809.5 tokens	50.1s	402.8 tokens
	Meta-learning Feedback	480.0s	4,909.5 tokens	20.6s	217.7 tokens
	Total Overhead	1,831.6s	18,257.7 tokens	123.2s	1,676.1 tokens

Table 7. Efficiency and Cost of Incremental Maintenance

temporal and factual contradictions that LLMs fail to capture. (2) *Robustness*. DocChecker achieves 76.81% intra- and 86.96% inter-document accuracy, vs. GPT-4o's 51.89% and 65.06% on ContraDoc. It attains near-perfect DSCD (0.992 binary, 0.962 evidence F_1). Graph fusion handles inter-document conflicts, where LLMs lag by 15.2%. (3) *Efficiency and scalability*. DocChecker scales linearly with document length (1k–5k tokens) and graph size (100–500 nodes). Parallelization reduces ContraDoc runtime from 7,410.11s to 974.44s (16 processors). In DSCD, DocChecker incurs minimal overhead (0.13s, 0.0 tokens), avoiding LLM costs in Tasks 2 and 3. Incremental maintenance cuts latency by 77.6%, updating rules in 4.3% of initial time. (4) *Ablation findings*. Graph Join boosts accuracy by 22.2%, while graph enrichment improves F1-score by 7.0%. Disabling ER raises FPR by 21.2% and lowers accuracy by 16.7%. Replacing LLM-based rule generation with a levelwise strategy reduces accuracy by 38.0%. In addition, removing meta-feedback lowers accuracy and F1-score by 16.6% and 19.9%, respectively. Eliminating verification/filtering further drops accuracy to 54.7%. These results highlight the importance of graph, ER and meta-learning components. (5) *Impact of prompts* (Appendices G.3–G.5 [1]). Even with SPO-link, CoVe, or counterfactual auditing, LLMs trail DocChecker by 22.19%–44.67%.

8 Conclusion

We propose graph-based RICOs and DocChecker framework, to detect contradictions within and across documents. By integrating graph analytics with learning-based rule discovery (Learner) and parallel detection (Detector), DocChecker effectively captures intra- and inter-document contradictions. Our experiments on real-world data verify that DocChecker are highly accurate and scalable. Future work includes (a) extending DocChecker to *cross-modal contradiction detection* (e.g., text-vision/tabular) via tractable multimodal RICOs, and (b) enhancing *robustness to LLM noise and hallucinations* through cross-model verification and consistency checks.

Acknowledgments

This work is supported by NSFC No. 62302503, Innovation Research Foundation of NUDT No. ZK23-15, the Open Research Fund from State Key Laboratory of High Performance Computing of China Grant No. 202401-09, and Young Elite Scientists Sponsorship Program by CAST No. YESS20240767.

References

- [1] 2026. Code, datasets and full version. <https://anonymous.4open.science/r/Doc4LLM-CE2E/>.
- [2] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
- [3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, and Shyamal Anadkat. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023). <https://arxiv.org/abs/2303.08774>
- [4] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 207–216.
- [5] João Paulo Aires and Felipe Meneguzzi. 2017. Norm Conflict Identification Using Deep Learning. In *AAMAS Workshops*. 194–207.
- [6] Abdulaziz Alamri. 2016. *The Detection of Contradictory Claims in Biomedical Abstracts*. Ph.D. Dissertation. University of Sheffield.
- [7] Abdulaziz Alamri and Mark Stevenson. 2016. A Corpus of Potentially Contradictory Research Claims from Cardiovascular Research. *Journal of Biomedical Semantics* 7 (2016), 36.
- [8] Alan Ansell, Felipe Bravo-Marquez, and Bernhard Pfahringer. 2021. PolyLM: Learning about Polysemy through Language Modeling. In *Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 563–574.
- [9] Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. Technical Report. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf
- [10] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *PODS*. 68–79.
- [11] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *international semantic web conference*. Springer, 722–735.
- [12] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, and Fei Huang. 2023. Qwen Technical Report. *arXiv preprint arXiv:2309.16609* (2023). <https://arxiv.org/abs/2309.16609>
- [13] Xianchun Bao, Zian Bao, Bie Binbin, QingSong Duan, Wenfei Fan, Hui Lei, Daji Li, Wei Lin, Peng Liu, Zhicong Lv, et al. 2024. Rock: Cleaning Data by Embedding ML in Logic Rules. In *Companion of the 2024 International Conference on Management of Data*. 106–119.
- [14] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150* (2020). <https://arxiv.org/abs/2004.05150>
- [15] Christopher M. Bishop and Nasser M. Nasrabadi. 2006. *Pattern Recognition and Machine Learning*. Springer.
- [16] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *EMNLP*. 632–642.
- [17] Hung-Ting Chen, Michael Zhang, and Eunsol Choi. 2022. Rich Knowledge Sources Bring Complex Knowledge Conflicts: Recalibrating Models to Reflect Conflicting Evidence. In *Conference on Empirical Methods in Natural Language Processing*. 2292–2307.
- [18] Yuhao Chen, Yuanjie Lyu, Shuo Chen Liu, Chao Zhang, Junhui Lv, and Tong Xu. 2025. Think Wider, Detect Sharper: Reinforced Reference Coverage for Document-Level Self-Contradiction Detection. In *EMNLP*. 1273–1288.
- [19] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and Jacob Devlin. 2024. Scaling Instruction-Finetuned Language Models. *JMLR* 25, 70 (2024), 1–53.
- [20] Mark Coddington and Logan Molyneux. 2023. When sources contradict: The epistemological functions of contradiction in news texts. *Journalism Studies* 24, 10 (2023), 1316–1333.
- [21] Cleo Condoravdi, Dick Crouch, Valeria De Paiva, Reinhard Stolle, and Daniel Bobrow. 2003. Entailment, Intensionality and Text Understanding. In *NAACL*. 38–45.
- [22] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. 2007. Improving Data Quality: Consistency and Accuracy. In *VLDB*. 315–326.
- [23] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges Workshop*. Springer, 177–190.
- [24] Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding Contradictions in Text. In *ACL. The Association for Computer Linguistics*, 1039–1047.
- [25] Tobias Deußer, Maren Pielka, Lisa Pucknat, Basil Jacob, Tim Dilmaghani, Mahdis Nourimand, Bernd Kliem, Rüdiger Loitz, Christian Bauckhage, and Rafet Sifa. 2023. Contradiction Detection in Financial Reports. In *Proceedings of the Northern Lights Deep Learning Workshop*.
- [26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.
- [27] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024.

Chain-of-verification reduces hallucination in large language models. In *Findings of the association for computational linguistics: ACL 2024*. 3563–3578.

- [28] Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmman, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 601–610.
- [29] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving Factuality and Reasoning in Language Models Through Multiagent Debate. In *International Conference on Machine Learning*.
- [30] David Easley and Jon Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Vol. 1. Cambridge university press Cambridge.
- [31] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitan-sky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024). <https://arxiv.org/abs/2404.16130>
- [32] Wenfei Fan, Wenzhi Fu, Ruochun Jin, Muyang Liu, Ping Lu, and Chao Tian. 2023. Making it tractable to catch duplicates and conflicts in graphs. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–28.
- [33] Wenfei Fan, Wenzhi Fu, Ruochun Jin, Ping Lu, and Chao Tian. 2022. Discovering Association Rules from Big Graphs. *PVLDB* 15, 7 (2022), 1479–1492.
- [34] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional functional dependencies for capturing data inconsistencies. *TODS* 33, 2 (2008), 6:1–6:48.
- [35] Wenfei Fan, Floris Geerts, Nan Tang, and Wenyuan Yu. 2014. Conflict resolution with data currency and consistency. *J. Data and Information Quality* 5, 1-2 (2014), 6:1–6:37.
- [36] Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Chao Tian, and Jingren Zhou. 2020. Capturing Associations in Graphs. *PVLDB* 13, 11 (2020), 1863–1876.
- [37] Wenfei Fan, Muyang Liu, Shuhao Liu, and Chao Tian. 2024. Capturing More Associations by Referencing External Graphs. *PVLDB* 17, 6 (2024), 1173–1186.
- [38] Wenfei Fan, Xueli Liu, Ping Lu, and Chao Tian. 2020. Catching Numeric Inconsistencies in Graphs. *TODS* 45, 2 (2020), 1–47.
- [39] Wenfei Fan, Ping Lu, and Chao Tian. 2020. Unifying Logic Rules and Machine Learning for Entity Enhancing. *Sci. China Inf. Sci.* 63, 7 (2020).
- [40] Wenfei Fan, Ping Lu, Chao Tian, and Jingren Zhou. 2019. Deducing Certain Fixes to Graphs. *PVLDB* 12, 7 (2019), 752–765.
- [41] Wenfei Fan and Chao Tian. 2022. Incremental graph computations: Doable and undoable. *ACM Transactions on Database Systems (TODS)* 47, 2 (2022), 1–44.
- [42] Wenfei Fan, Chao Tian, Yanghao Wang, and Qiang Yin. 2021. Parallel Discrepancy Detection and Incremental Detection. *PVLDB* 14, 8 (2021), 1351–1364.
- [43] Wenfei Fan, Chao Tian, Ruiqi Xu, Qiang Yin, Wenyuan Yu, and Jingren Zhou. 2021. Incrementalizing graph algorithms. In *Proceedings of the International Conference on Management of Data*. 459–471.
- [44] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional dependencies for graphs. In *Proceedings of the International Conference on Management of Data*. 1843–1857.
- [45] Anna Fariha, Ashish Tiwari, Alexandra Meliou, Arjun Radhakrishna, and Sumit Gulwani. 2021. CoCo: Interactive Exploration of Conformance Constraints for Data Understanding and Data Cleaning. In *SIGMOD*. ACM, 2706–2710.
- [46] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.
- [47] Matthias Feurer, Aaron Klein, Katharina Eggersperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *NeurIPS*. 2962–2970.
- [48] Lampros Flokas, Weiyan Wu, Yejia Liu, Jiannan Wang, Nakul Verma, and Eugene Wu. 2022. Complaint-Driven Training Data Debugging at Interactive Speeds. In *SIGMOD*. 369–383.
- [49] Etienne Fortier-Dubois and Domenic Rosati. 2023. Using Contradictions Improves Question Answering Systems. In *ACL*. 827–840.
- [50] Michael Garey and David Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.
- [51] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.
- [52] Rihan Hai, Christos Koutras, Andra Ionescu, Ziyu Li, Wenbo Sun, Jessie van Schijndel, Yan Kang, and Asterios Katsifodimos. 2023. Amalur: Data Integration Meets Machine Learning. In *ICDE*. IEEE, 3729–3739.
- [53] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, Vol. 30. 1025–1035.
- [54] Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI*, Vol. 6. 755–762.

- [55] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543* (2021).
- [56] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. In *SIGMOD*. 829–846.
- [57] Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing Textual Entailment with LCC’s GROUNDHOG System. In *The Second PASCAL Challenges Workshop*, Vol. 18. 1–4.
- [58] Benjamin Hilprecht, Christian Hammacher, Eduardo Souza dos Reis, Mohamed Abdelaal, and Carsten Binnig. 2023. DiffML: End-to-end Differentiable ML Pipelines. In *DEEM@SIGMOD*. ACM, 7:1–7:7.
- [59] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *EMNLP*. 782–792.
- [60] Cheng Hsu, Cheng-Te Li, Diego Saez-Trumper, and Yi-Zhan Hsu. 2021. Wikicontradiction: Detecting Self-Contradiction Articles on Wikipedia. In *IEEE International Conference on Big Data*. 427–436.
- [61] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HJIWWSFDH>
- [62] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *The Web Conference 2020*. 2704–2710.
- [63] Liusheng Huang, Huaping Chen, Xun Wang, and Guoliang Chen. 2000. A fast algorithm for mining association rules. *Journal of Computer Science and Technology* 15, 6 (2000), 619–624.
- [64] Pengcheng Jiang, Lang Cao, Ruike Zhu, Minhao Jiang, Yunyi Zhang, Jimeng Sun, and Jiawei Han. 2025. RAS: Retrieval-And-Structuring for Knowledge-Intensive LLM Generation. *CoRR* abs/2502.10996 (2025). <https://doi.org/10.48550/arXiv.2502.10996>
- [65] Cheng Jiayang, Chunkit Chan, Qianqian Zhuang, Lin Qiu, Tianhang Zhang, Tengxiao Liu, Yangqiu Song, Yue Zhang, Pengfei Liu, and Zheng Zhang. 2024. ECON: On the Detection and Resolution of Evidence Conflicts. In *Conference on Empirical Methods in Natural Language Processing*. 7816–7844.
- [66] Di Jin, Sijia Liu, Yang Liu, and Dilek Hakkani-Tur. 2022. Improving Bot Response Contradiction Detection via Utterance Rewriting. In *SIGDIAL*. 605–614.
- [67] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International conference on machine learning*. PMLR, 15696–15707.
- [68] Bojan Karlaš, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. 2020. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *arXiv preprint arXiv:2005.05117* (2020).
- [69] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.
- [70] Yuta Koreeda and Christopher D. Manning. 2021. ContractNLI: A Dataset for Document-Level Natural Language Inference for Contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 1907–1919.
- [71] Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, and Eugene Wu. 2017. BoostClean: Automated Error Detection and Repair for Machine Learning. *CoRR* abs/1711.01299 (2017).
- [72] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. ActiveClean: Interactive Data Cleaning For Statistical Modeling. *PVLDB* 9, 12 (2016), 948–959.
- [73] Sanjay Krishnan and Eugene Wu. 2019. AlphaClean: Automatic Generation of Data Cleaning Pipelines. *CoRR* abs/1904.11827 (2019). <http://arxiv.org/abs/1904.11827>
- [74] Clyde P. Kruskal, Larry Rudolph, and Marc Snir. 1990. A Complexity Theory of Efficient Parallel Algorithms. *Theoretical Computer Science* 71, 1 (1990), 95–132.
- [75] Piroska Lendvai, Isabelle Augenstein, Kalina Bontcheva, and Thierry Declerck. 2016. Monolingual Social Media Datasets for Detecting Contradiction and Entailment. In *LREC*. 4602–4605.
- [76] Piroska Lendvai and Uwe Reichel. 2016. Contradiction Detection for Rumorous Claims. In *The Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics (ExProM)*. 31–40.
- [77] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. In *ACL*. 7871–7880.
- [78] Chuqin Li, Xi Niu, Ahmad Al-Doulat, and Noseong Park. 2018. A Computational Approach to Finding Contradictions in User Opinionated Text. In *ASONAM*. 351–356.
- [79] Jierui Li, Vipul Raheja, and Dhruv Kumar. 2024. ContraDoc: Understanding Self-Contradictions in Documents with Large Language Models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 6509–6523.

- [80] Peng Li, Zhiyi Chen, Xu Chu, and Kexin Rong. 2023. DiffPrep: Differentiable Data Preprocessing Pipeline Search for Learning over Tabular Data. *Proc. ACM Manag. Data* 1, 2 (2023), 183:1–183:26.
- [81] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2021. CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks. In *ICDE. IEEE*, 13–24.
- [82] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, and Chong Ruan. 2024. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437* (2024). <https://arxiv.org/abs/2412.19437>
- [83] Hanmeng Liu, Leyang Cui, Jian Liu, and Yue Zhang. 2021. Natural language inference in context-investigating contextual reasoning over long texts. In *AAAI*, Vol. 35. 13388–13396.
- [84] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranajpe, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics* 12 (2024), 157–173.
- [85] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019). <https://arxiv.org/abs/1907.11692>
- [86] Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. 2020. Picket: Self-supervised Data Diagnostics for ML Pipelines. *CoRR abs/2006.04730v1* (2020). <https://arxiv.org/abs/2006.04730v1>
- [87] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-Based Knowledge Conflicts in Question Answering. In *Conference on Empirical Methods in Natural Language Processing*. 7052–7063.
- [88] Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to Recognize Features of Valid Textual Entailments. In *ACL*. 41–48.
- [89] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems* 36 (2023), 46534–46594.
- [90] Mohammad Mahdavi, ZiaWasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A Configuration-Free Error Detection System. In *SIGMOD*. 865–882.
- [91] Stanislav Malyshev, Markus Kröttsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. 2018. Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia’s Knowledge Graph. In *ISWC*. 376–394.
- [92] Puneet Mathur, Gautam Kunapuli, Riyaz Bhat, Manish Shrivastava, Dinesh Manocha, and Maneesh Singh. 2022. DocInfer: Document-level natural language inference using optimal evidence selection. In *EMNLP*. 809–824.
- [93] Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, and Heiner Stuckenschmidt. 2020. Reinforced Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. *arXiv preprint arXiv:2004.04412* (2020). <https://arxiv.org/abs/2004.04412>
- [94] Roser Morante and Caroline Sporleder. 2012. Modality and Negation: An Introduction to the Special Issue. *Comput. Linguist.* 38, 2 (2012), 223–260.
- [95] Yixin Nie, Mary Williamson, Mohit Bansal, Douwe Kiela, and Jason Weston. 2021. I Like Fish, Especially Dolphins: Addressing Contradictions in Dialogue Modeling. In *ACL*. 1699–1713.
- [96] OpenAI. 2025. GPT-5.2 Technical Report. <https://openai.com/index/introducing-gpt-5-2/>
- [97] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *TKDE* 36, 7 (2024), 3580–3599.
- [98] James Pustejovsky, José M. Castano, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003. TimeML: Robust Specification of Event and Temporal Expressions in Text. *New Directions in Question Answering* 3 (2003), 28–34.
- [99] ABM Ashikur Rahman, Saeed Anwar, Muhammad Usman, Irfan Ahmad, and Ajmal Mian. 2025. DefAn: Definitive Answer Dataset for LLM Hallucination Evaluation. *Information* 16, 11 (2025), 937.
- [100] Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine Reading Comprehension with Numerical Reasoning. In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2474–2484.
- [101] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *PVLDB* 11, 3 (2017), 269–282.
- [102] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*. 3980–3990.
- [103] Graciela Rosemblat, Marcelo Fiszman, Dongwook Shin, and Halil Kilicoglu. 2019. Towards a Characterization of Apparent Contradictions in the Biomedical Literature Using Context Analysis. *JBI* 98 (2019), 103275.
- [104] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a Distilled Version of BERT:

- Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108* (2019). <https://arxiv.org/abs/1910.01108>
- [105] Farzaneh Sarafraz. 2012. *Finding Conflicting Statements in the Biomedical Literature*. Ph. D. Dissertation. The University of Manchester.
 - [106] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*. 593–607.
 - [107] Tal Schuster, Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, and Donald Metzler. 2022. Stretching Sentence-pair NLI Models to Reason over Long Documents and Clusters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 394–412.
 - [108] Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. 2022. Neural Entity Linking: A Survey of Models Based on Deep Learning. *Semantic Web* 13, 3 (2022), 527–570.
 - [109] Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *TKDE* 27, 2 (2014), 443–460.
 - [110] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. *NeurIPS* 36 (2023), 8634–8652.
 - [111] Zhaochen Su, Zecheng Tang, Xinyan Guan, Lijun Wu, Min Zhang, and Juntao Li. 2022. Improving Temporal Generalization of Pre-Trained Language Models with Lexical Semantic Change. In *EMNLP*. 6380–6393.
 - [112] Zhaochen Su, Jun Zhang, Xiaoye Qu, Tong Zhu, Yanshu Li, Jiahuo Sun, Juntao Li, Min Zhang, and Yu Cheng. 2024. ConflictBank: A Benchmark for Evaluating the Influence of Knowledge Conflicts in LLMs. In *NeurIPS*, Vol. 37. 103242–103268.
 - [113] Melanie Subbiah, Faisal Ladhak, Akanksha Mishra, Griffin Thomas Adams, Lydia Chilton, and Kathleen McKeown. 2024. StorySumm: Evaluating Faithfulness in Story Summarization. In *EMNLP*. 9988–10005.
 - [114] Katia P. Sycara. 1993. Machine learning for intelligent support of conflict resolution. *Decision Support Systems* 10, 2 (1993), 121–136.
 - [115] Ki Hyun Tae, Yuji Roh, Young Hun Oh, Hyunsu Kim, and Steven Euijong Whang. 2019. Data Cleaning for Accurate, Fair, and Robust Models: A Big Data - AI Integration Approach. In *DEEM@SIGMOD*. ACM, 5:1–5:4.
 - [116] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).
 - [117] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 809–819.
 - [118] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, and Shruti Bhosale. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024). arXiv:2407.21783 [cs.AI] <https://arxiv.org/abs/2407.21783>
 - [119] Leslie G. Valiant. 1990. A Bridging Model for Parallel Computation. *Commun. ACM* 33, 8 (1990), 103–111.
 - [120] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903 [stat.ML] <https://arxiv.org/abs/1710.10903>
 - [121] Larysa Visengeriyeva and Ziawasch Abedjan. 2018. Metadata-driven error detection. In *SSDBM*. 1:1–1:12.
 - [122] Denny Vrandečić. 2012. Wikidata: A New Platform for Collaborative Data Collection. In *International Conference on World Wide Web*. 1063–1064.
 - [123] Alexander Wan, Eric Wallace, and Dan Klein. 2024. What Evidence Do Language Models Find Convincing? arXiv:2402.11782 [cs.CL] <https://arxiv.org/abs/2402.11782>
 - [124] Keheng Wang, Feiyu Duan, Peiguang Li, Sirui Wang, and Xunliang Cai. 2025. LLMs Know What They Need: Leveraging a Missing Information Guided Framework to Empower Retrieval-Augmented Generation. In *COLING*. 2379–2400.
 - [125] Yike Wang, Shangbin Feng, Heng Wang, Weijia Shi, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024. Resolving Knowledge Conflicts in Large Language Models. arXiv:2310.00935 [cs.CL] <https://arxiv.org/abs/2310.00935>
 - [126] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *ACL*. 13484–13508.
 - [127] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers)*. 1112–1122.
 - [128] Weiyuan Wu, Lampros Flokas, Eugene Wu, and Jiannan Wang. 2020. Complaint-driven Training Data Debugging for Query 2.0. In *SIGMOD*. ACM, 1317–1334.
 - [129] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2023. Adaptive Chameleon or Stubborn Sloth: Revealing the Behavior of Large Language Models in Knowledge Conflicts. In *ICLR*.
 - [130] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Conference on Empirical Methods in Natural Language Processing*. 564–573.

- [131] Haike Xu, Zongyu Lin, Yizhou Sun, Kai-Wei Chang, and Piotr Indyk. 2024. SparseCL: Sparse Contrastive Learning for Contradiction Retrieval. *arXiv:2406.10746 [cs.CL]* <https://arxiv.org/abs/2406.10746>
- [132] Rongwu Xu, Brian Lin, Shujian Yang, Tianqi Zhang, Weiyan Shi, Tianwei Zhang, Zhixuan Fang, Wei Xu, and Han Qiu. 2024. The Earth Is Flat Because...: Investigating LLMs' Belief Towards Misinformation via Persuasive Conversation. In *ACL*. 16259–16303.
- [133] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *NeurIPS* 32 (2019).
- [134] Wenpeng Yin, Dragomir Radev, and Caiming Xiong. 2021. DocNLI: A Large-Scale Dataset for Document-Level Natural Language Inference. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 4913–4922.
- [135] Zhangdie Yuan and Andreas Vlachos. 2023. Zero-Shot Fact-Checking with Semantic Triples and Knowledge Graphs. *CoRR abs/2312.11785* (2023).
- [136] Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2007. Shallow Semantic in Fast Textual Entailment Rule Learners. In *ACL*. 72–77.
- [137] Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. GLiNER: Generalist Model for Named Entity Recognition Using Bidirectional Transformer. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 5364–5376.
- [138] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-Bert: Only Attention is Needed for Learning Graph Representations. (2020). *arXiv:2001.05140 [cs.LG]* <https://arxiv.org/abs/2001.05140>
- [139] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2025. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *Computational Linguistics* 51, 4 (2025), 1373–1418.
- [140] Chujie Zheng, Jinfeng Zhou, Yinhe Zheng, Libiao Peng, Zhen Guo, Wenquan Wu, Zheng-Yu Niu, Hua Wu, and Minlie Huang. 2022. CDConv: A Benchmark for Contradiction Detection in Chinese Conversations. In *EMNLP*. 18–29.
- [141] Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When Does Pretraining Help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset of 53,000+ Legal Holdings. In *International Conference on Artificial Intelligence and Law*. 159–168.

Received 17 January 2026; revised 19 May 2026; accepted 12 June 2026